


MULTIBUS<sup>®</sup> II BUS  
ARCHITECTURE SPECIFICATION  
HANDBOOK



\$25.00





 quantum electronics  
Box 391262  
Bramley  
2018

# MULTIBUS® II BUS ARCHITECTURE SPECIFICATION

## Chapter I Introduction to the Intel® MULTIBUS® II Bus Architecture

I

## Chapter II Intel® iPSB™ Bus Specification

II

## Chapter III Intel® iLBX™ II Bus Specification

III

## Chapter IV Intel® iSSB™ Bus Specification

IV

## Chapter V Intel® MULTIBUS® II Mechanical Specification

V

## Chapter VI Intel® MULTIBUS® II System Interface Specification

VI



REV.	REVISION HISTORY	PRINT DATE
- A	Original Issue.	10/83
- B	iLBX™ II Bus Specification and System Interface Specification added.	11/83

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department  
Intel Corporation  
3065 Bowers Avenue  
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9(a)(9).

No part of this document may be copied or reproduced in any form or by any means without prior written consent of Intel Corporation.

Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and its affiliates and may be used only to identify Intel products:

BITBUS	iLBX	iPDS	MULTICHANNEL
COMMputer	im	iPSB	MULTIMODULE
CREDIT	iMMX	iRMX	Plug-A-Bubble
Data Pipeline	Insite	iSBC	PROMPT
GENIUS	Intel	iSBX	Promware
Δ	intel	iSDM	QUEX
i	intelBOS	iSSB	QUEST
i <sup>2</sup> ICE	Intelevision	iSXM	Ripplemode
ICE	Intelligent Identifier	Library Manager	RMX/80
iCS	Intelligent Programming	MCS	RUPI
iDBP	Intellec	Megachassis	Seamless
iDIS	Intellink	MICROMAINFRAME	SOLO
	iOSP	MULTIBUS	SYSTEM 2000
			UPI

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

Patent Pending

Copyright © 1983; Intel Corporation





# **INTRODUCTION TO THE INTEL MULTIBUS® II SYSTEM ARCHITECTURE**

---





---

# MULTIBUS® II SYSTEM ARCHITECTURE INTRODUCTION TO THE INTEL

---







## CONTENTS

	PAGE
INTRODUCTION TO THE MULTIBUS® II BUS ARCHITECTURE	
1.1 Architecture Overview.....	1-1
1.1.1 An Answer to Obsolescence.....	1-2
1.1.2 Multiple Bus Structure.....	1-2
1.1.3 Multiple Bus Structures Enhance Functional Partitioning.....	1-3
1.2 The MULTIBUS® II Busses.....	1-4
1.2.1 Parallel System Bus (iPSB™ Bus).....	1-4
1.2.2 Local Bus Extension (iLBX™ II).....	1-6
1.2.3 Serial System Bus (iSSB™).....	1-7
1.2.4 Multichannel™ DMA I/O Bus.....	1-8
1.2.5 iSBX™ I/O Expansion Bus.....	1-8
1.3 Building a Computer System for the Future.....	1-8
1.3.1 Increased CPU Demand.....	1-9
1.3.2 Increased Data Movement.....	1-10
1.3.3 Lower Cost Requirements.....	1-11
1.4 Summary.....	1-12

### FIGURES

1-1. MULTIBUS® II Bus Architecture (5 Busses).....	1-1
1-2. System Providing Basic Requirements.....	1-9
1-3. System With Increased CPU Requirements.....	1-10
1-4. Enhanced Support of Multiple Processors.....	1-11
1-5. Low-Cost System Design.....	1-12





PAGE	
	INTRODUCTION TO THE MULTIBUS <sup>®</sup> II BUS ARCHITECTURE
1-1	Architecture Overview.....
1-1.1	An Answer to Obsolescence.....
1-1.1.1	Multibus Bus Structures.....
1-1.1.1.1	Multibus Bus Structures Enhance Functional Partitioning.....
1-2	The MULTIBUS <sup>®</sup> II Bus.....
1-2.1	Parallel System Bus (PSB) Bus.....
1-2.2	Local Bus Extension (LBE) II.....
1-2.3	Serial System Bus (SSB).....
1-2.4	"Multichannel" DMA I/O Bus.....
1-2.5	"SSB" I/O Expansion Bus.....
1-3	Building a Computer System for the Future.....
1-3.1	Increased CPU Demand.....
1-3.2	Increased Data Movement.....
1-3.3	Lower Cost Requirements.....
1-4	Summary.....

## FIGURES

1-1	MULTIBUS <sup>®</sup> II Bus Architecture (2 buses).....
1-2	System Providing Basic Requirements.....
1-3	System With Increased CPU Requirements.....
1-4	Enhanced Support of Multiple Processors.....
1-5	Low-Cost System Design.....



# CHAPTER 1 INTRODUCTION

## 1.1 ARCHITECTURE OVERVIEW

The Multibus II bus architecture is an advanced, processor-independent, open system architecture suitable for a wide range of microprocessor-based designs. The multiple bus architecture includes three bus structures defined in this specification and compatibility with two existing Multibus I/O busses. Multibus II systems offer designers significant performance advantages and advanced features including a 32-bit parallel system bus with 40M byte/sec throughput, high-speed access to large amounts of off-board memory, a low-cost serial system bus, and effective multiprocessor support.

The Multibus II bus architecture consists of the Parallel System (iPSB) Bus, the Local Bus Extension (iLBX II Bus), the Serial System (iSSB) Bus, and two busses carried over from the Multibus I architecture -- the iSBX I/O Expansion Bus and the Multichannel DMA (Direct Memory Access) I/O Bus (Figure 1-1). A common system interface which defines intermodule communication and data transfer protocols ties the busses together and allows designers to choose from several combinations of the five to meet specific application requirements.

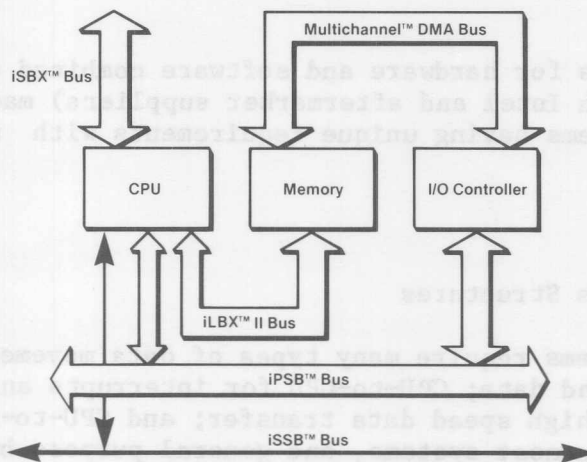


Figure 1-1. MULTIBUS® II Bus Architecture (5 Busses)



## INTRODUCTION TO THE MULTIBUS® II BUS ARCHITECTURE

Because of its multiple bus structure and its ability to support 32-bit microprocessors, the Multibus II bus architecture provides an evolutionary path to both future system expansion and future VLSI technology. The architectures supported by the Multibus II system have migrated from the Multibus I system.

### 1.1.1 An Answer To Obsolescence

In the last decade, the avalanche of new microcomputer technology, especially VLSI, threatened to obsolete products almost before they went into production. To buffer user's from this onrush of technology, Intel helped develop standard interfaces. One of the most notable was the Multibus I bus architecture.

The Multibus I interface became not only the industry standard, but it was designated the IEEE 796 standard as well. With the Multibus I interface, user's could exploit the benefits of VLSI technology without having to pay a premium for new system design.

Other benefits from the use of standard interfaces in the Multibus I architecture soon followed. As Intel's "Open Systems" design strategy gained wide acceptance, aftermarket support grew, providing multiple supply sources, wide product selections and competitive prices. Today, 200 companies manufacture over 1,200 Multibus I compatible products.

The Open Systems approach was also demonstrated in Intel products which provide compatible products at three levels of integration: components, boards, and systems. This multilevel approach has enabled OEMs to adapt to new business environments and opportunities as VLSI technology expanded.

Standard interfaces for hardware and software combined with many Multibus products (from both Intel and aftermarket suppliers) made it possible to configure new systems having unique requirements with minimal risk and investment.

### 1.1.2 Multiple Bus Structures

Microcomputer systems require many types of data movement: memory-to-CPU for instructions and data; CPU-to-CPU for interrupts and messages; I/O-to-memory for high speed data transfer; and CPU-to-I/O for direct control of I/O. In most systems, one general purpose bus can do all these three functions. However, for high performance systems, a general purpose bus lacks the total bandwidth required. And, in low-cost systems, the general purpose bus may be too costly.

A multiple bus structure provides specialized busses for specific critical functions. Four important advantages result.

1. The bandwidth of the general purpose bus is preserved, creating a "virtual bandwidth" for interprocessor communication and data movement.
2. The specialized bus does its function better than a general purpose bus.
3. Functions can be carried out in parallel on different busses.
4. Users can tailor their system performance and avoid unnecessary costs by selecting only those busses required for their application.

While the Multibus I interface pioneered the multiple bus approach, the Multibus II bus architecture refines it and extends its range. The new architecture offers five processor-independent busses which give system designers the ability to configure their systems for their needs today as well as meeting the future system requirements.

Each of the Multibus II interfaces is fully compatible with the others. Thus, in general, it is simply a matter of choosing the appropriate bus or combination of busses to fit the exact needs of a particular application. Moreover, the standard interfaces mean designers can reconfigure the system as new requirements demand -- or as VLSI technology provides improvements in microprocessor technology.

Multibus I users can upgrade to the Multibus II architecture as their needs and bandwidth expand to 32-bit capabilities, or as their 16-bit systems begin to require more sophisticated multiprocessing capability. Therefore, new users requiring high performance 16- or 32-bit data paths optimized for multiprocessing will find the multiple bus structure of the Multibus II bus architecture ideal for advanced microprocessor design.

#### 1.1.3 Multiple Bus Structures Enhance Functional Partitioning

Each multiple bus structure is tailored for a particular purpose. This is part of a design philosophy called "Functional Partitioning." Basically, functional partitioning is a modular design approach that requires breaking an overall problem into manageable pieces based on function. For example, typical microcomputer system functions are mass storage control, data processing, communications and graphics.

In typical functionally partitioned systems, data movement between agents is minimized. Requests for data movement are kept to a minimum, and critical real-time data should be kept in the local environment. Once the agents have been defined, they are implemented by optimizing each for its specific requirement. The Multibus II bus architecture defines standard interfaces between each functional module and tailors each interface to its specific function.

## INTRODUCTION TO THE MULTIBUS® II BUS ARCHITECTURE

For example, the Parallel System Bus (iPSB bus) is optimized for interprocessor communication and data movement. The Local Bus Extension, (iLBX II bus) is similarly optimized for very high speed execution. And the Serial System Bus (iSSB bus) is optimized for low-cost interprocessor communication.

Thus, the Multibus II bus architecture provides the means to design a system optimized for performance with each bus serving a specialized function. Since each bus is also optimized for performance, functional partitioning of the agents is supported and enhanced.

### 1.2 THE MULTIBUS® II BUSES

The Multibus II bus architecture consists of the Parallel System Bus, the Local Bus Extension, the Serial System Bus, and two busses carried over from the Multibus I architecture -- the iSBX I/O Expansion Bus and the Multichannel DMA (Direct Memory Access) I/O Bus.

#### 1.2.1 Parallel System Bus (iPSB™ Bus)

The Multibus II Parallel System Bus is a high-performance, general purpose bus that provides important data movement and inter-processor communication functions. Being general purpose in nature, the iPSB bus also supports arbitration, execution and I/O data movement and board configuration support.

The iPSB bus supports four address spaces: a 32-bit memory address space, a 16-bit I/O address space, an 8-bit message address space, and a 16-bit interconnect address space. Data is clocked at 10 MHz and can be up to 32 bits wide.

In addition, the iPSB bus incorporates features which:

- o Provide high-performance data movement
- o Enhance multiprocessor support
- o Improve ease-of-use
- o Increase system reliability
- o Bounded real-time response

The following is a brief discussion of those features.

#### High Performance

The Parallel System Bus has a burst transfer capability yielding a maximum sustained bandwidth of 40 megabytes/second. The burst is implemented as a single address cycle followed by multiple data transfers which maximize the bus bandwidth.



Message Passing is another important attribute of the Parallel System Bus. This feature allows two bus agents (i.e., boards) to exchange information in blocks of data. The iPSB bus method of message passing provides a high performance facility for moving data from one functional module to another without having to worry about memory management or synchronization problems at the bus interface.

The iPSB bus supports multiple processor agents. That is, the arbitration features can support up to 20 requests for the bus at the same time. This supports the functional partitioning approach and maximizes the effectiveness of each function.

#### Ease of Use

The iPSB bus is a processor-independent general purpose system bus. As such, it supports 8-, 16-, and 32-bit processors, and is capable of transferring 8, 16, 24, or 32 bits of data. This attribute makes it extremely flexible.

General system-wide functions such as power-up/power-fail, bus time-out, system timing references, time-of-day clock, etc. are centralized into one module called the Central Services Module. In multiple agent systems, centralization of such functions reduces system cost, frees board area for other functions and is generally more efficient. The Central Services Module can be located on a unique board dedicated to that function.

The Interconnect Address Space is provided for dynamic configuration of Multibus II boards. This feature allows board options to be programmed or read from interconnect space which simplifies the configuration task and allows system resources to be identified. It also allows system-level diagnostics to access the results of board-level diagnostics for system confidence reports.

#### Reliability

Since the Parallel System Bus is synchronous, the signals only have to be valid at specific intervals. Thus, noise-induced signals are not likely to be erroneously interpreted as data or control.

Reliability is also enhanced by such features as parity on transfers, DIN connectors and distributed ground pins. Byte parity is checked and generated for address, data and command lines. The pin and socket design of DIN connectors makes them extremely reliable. Also, the isolation of critical signals with adjacent ground pins reduces noise in nearby circuits on the backplane.

## INTRODUCTION TO THE MULTIBUS® II BUS ARCHITECTURE

In summary, the Parallel System Bus is a full-functioning, general purpose system bus which enhances multiprocessor support by its unique message passing facility. It improves ease-of-use through its multiple data width support and interconnect facilities. Finally, the iPSB bus increases system reliability by virtue of its synchronous nature, its parity on transfers, and its two-piece DIN connectors.

### 1.2.2 Local Bus Extension (iLBX™ II)

Multiple CPUs executing instructions in shared global memory can easily saturate the system bus, resulting in overall system performance degradation. An extension of the on-board processor bus provides the means to remove the processor execution functions from the general purpose system bus and extend local on-board processor bus provides the means to remove the processor execution functions from the general purpose system bus and extend local on-board performance capability to off-board memory resources.

The iLBX II Bus helps to maximize performance in Multibus II systems by providing arbitration-free local memory expansion to 64 megabytes and a maximum bus clock rate of 12 MHz. It is a processor-independent bus that supports 8-, 16-, and 32-bit processors and up to six agents or boards.

The iLBX II Bus provides an alternate very high bandwidth path (48 megabyte/sec) to the processor's memory resources. In doing so, it reduces the processor's system bus bandwidth requirements by 60%-90%. Since it has been optimized for execution, it does not provide the functions of I/O or message passing.

A synchronous bus providing increased reliability, the iLBX II bus incorporates a number of advanced features which enhance system performance. For example, the iLBX II bus allows pipelining; the ability for the address portion of the following cycle to overlap on the data portion of the current cycle. Pipelining promotes the efficient use of the execution bus and helps make possible its higher bandwidth.

Another contributing factor to the higher bandwidth on the iLBX II bus is its ability to support block transfers. With one address phase, the bus can obtain multiple pieces of data, again a more efficient use of the execution bus.

As with the iLBX bus in the Multibus I architecture, more than one iLBX II bus may exist in a Multibus II system to optimize each processor's performance.

The iLBX II bus, then, is a high-speed, high-bandwidth execution bus which extends the on-board local bus to off-board memory resources. Its 48 megabytes/sec bandwidth, bus clock rate of 12MHz and advanced features like pipelining and block transfers make it a high performance, low latency, reliable bus for today's multiprocessor architectures.

### 1.2.3 Serial System Bus (iSSB™)

The Serial System Bus is very closely tied to some characteristics of the Parallel System Bus since it implements the message passing functions of the iPSB bus with a low-cost serial interconnect. That is, the iSSB bus is a low-cost alternative to the message interface on the iPSB bus.

Whereas the iPSB bus 32-bit wide parallel transfers and runs at 10 MHz, the iSSB bus is 1-bit wide and runs at 2 MHz. Thus, the performance is roughly 2 orders of magnitude less. However, the cost is also 2 orders of magnitude less.

There are two cost-reduction mechanisms involved. The first is the interface device. While the iPSB bus has 96 pins, the iSSB bus has only two.

The second consideration is packaging flexibility. The electrical requirements of the iPSB bus require it to be implemented in a very restrictive manner, with boards stacked and cables run to each device being controlled. On the other hand, the iSSB bus may be extended a distance of 10 meters, allowing boards to be scattered within a box or even be located in separate boxes. As VLSI devices become more capable, this will actually eliminate boards within a system by allowing the bus to be extended to the point of control. Thus, the iSSB bus has the ability to be physically distributed since it is no longer restricted to the backplane. However, it will likely remain inside the physical packaging of the system, even though it may extend short distances to connect a modular package.

As VLSI technology continues to shrink more and more functions into a single piece of silicon, the printed circuit board area needed to implement a function will be decreased. In fact, today the interface to the iSSB bus can be implemented with a single chip. This level of integration will provide the lowest possible system cost. Further, the controller function found in contemporary systems will migrate to the printed circuit boards of the peripherals. The iSSB bus, then, allows reduced interconnect costs and physical distribution of system agents.

While taking advantage of the cost and distribution of a serial media, the iSSB bus allows functional agents to communicate with an identical software message passing interface to that of the Parallel System Bus. This means that agents that communicate via message passing on the iPSB bus can migrate easily to an iSSB bus with only minimal software changes.

Whereas the iPSB bus in combination with future VLSI advances will offer increased capabilities at the same cost, the iSSB bus in combination with VLSI technology will offer the same capabilities at a lower cost.



#### 1.2.4 Multichannel™ DMA I/O Bus

Carried over from the Multibus I bus architecture, the Multichannel I/O Bus solves the problem of high-speed I/O data to and from physically distributed custom peripherals such as mass storage devices or graphics display systems. The Multichannel bus allows high-speed (8 megabytes/sec) block transfers of data over the data path between such peripherals and single board computers.

The Multichannel bus provides a standardized I/O interface with full speed operation at up to 15 meters with a simple asynchronous protocol. It supports up to 16 devices, both 8- and 16-bit and provides 16 megabyte memory or register address space per device. Typical uses of this bus include computer graphics, specialized peripheral control, data acquisition and high-speed Multibus system-to-system communication.

#### 1.2.5 iSBX™ I/O Expansion Bus

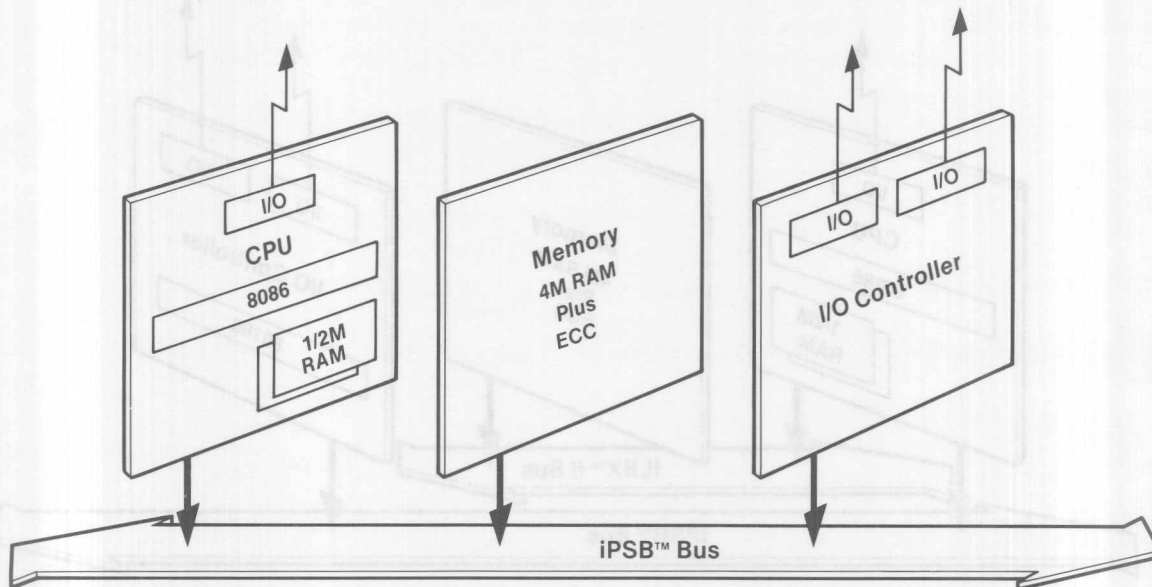
Also a carryover from the Multibus I bus architecture, the iSBX I/O expansion bus allows incremental on-board system expansion through the use of small iSBX Multimodule boards. Currently, iSBX boards add capability in the areas of parallel I/O, serial I/O graphics, bubbles and advanced mathematics functions. All iSBX boards afford system expansion without the additional cost of adding another full expansion board.

The expansion bus allows users to customize their single board computers to individual applications in response to latest VLSI technology. Since they are able to buy exactly the capabilities needed, both system cost and system size are kept to a minimum.

### 1.3 BUILDING A COMPUTER SYSTEM FOR THE FUTURE

The advantages of the multiple bus approach of Multibus II bus architecture is easily demonstrated by a simple system that evolves over time to a family of products. Suppose the basic requirement is for moderate CPU power, variable sizes of RAM depending on system usage, and a variety of simple I/O devices.

The initial system may contain a sixteen bit microprocessor like the 8 MHz 8086 which provides the CPU requirements (Figure 1-2). Considerable RAM can be placed on the same board as the CPU: up to one megabyte with 256K DRAMS and a RAM expansion Multimodule which mechanically does not require a second slot. The CPU board could still accommodate extensive I/O facilities such as serial, parallel, or analog I/O, as well as iSBX connectors for low-cost I/O additions to the base board.



- Execution on the iPSB™ Bus Via Memory Address Space
- I/O Control on the iPSB™ Bus Via I/O Address Space
- System Configuration Via Interconnect Space

x-568

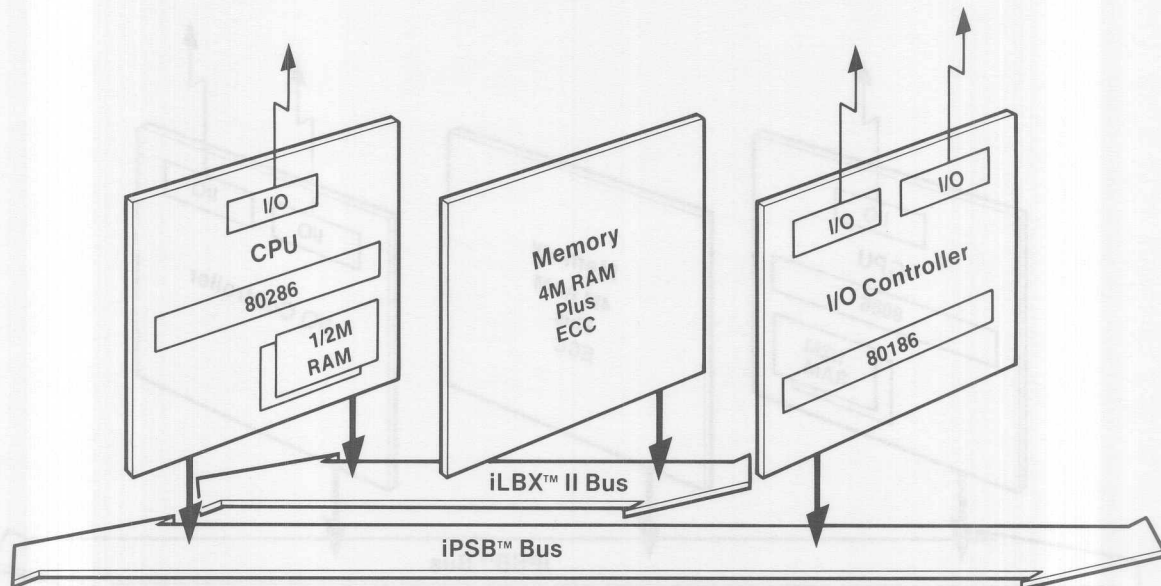
Figure 1-2. System Providing Basic Requirements

Additional RAM can be added with memory boards on the Parallel System Bus with capacities up to 2 megabytes with 256K DRAMS and features such as parity or ECC. Additional I/O can be added on boards accessed via the Parallel System Bus with software compatibility with the on-board I/O. In both cases, the I/O is addressed via the I/O space of the 8086 since the iPSB bus supports an equivalent I/O space.

Finally, diagnostic and system start-up code can be written which utilizes the interconnect space to dynamically determine the features of the boards in the system. Thus, customization and incremental enhancement of the computer for each customer is simplified, resulting in greater customer satisfaction.

### 1.3.1 Increased CPU Demand

In time the computer will be expected to perform increased workloads as competitive pressures demand productivity improvements from the user. The processor board could be designed to use an 8 MHz 80286 processor (Figure 1-3). Alternatively, a second processor board could be added.



- Processor Execution on the iLBX™ II Bus

x-569

Figure 1-3. System With Increased CPU Requirements

To save system bus bandwidth and expand performance with large memory sizes, the new processor board and memory could be designed with iLBX II bus interfaces. The two board set, connected via the iLBX II bus, will function as a virtual single board, independent of any other sets of iLBX II connected boards.

### 1.3.2 Increased Data Movement

With an increased demand for services of the I/O, preprocessing of the data will be required to reduce the information flow to only the relevant data. This can be accomplished by adding processor power to the I/O interfaces. As standard interfaces are desirable to preserve the software investment, the simple I/O address mapping of the example would be replaced with a higher level procedural interface. This higher level would be provided by software for on-board I/O. The off-board I/O, however, is more complicated.

The data could be moved between processors via shared memory, but this would require the addition of a memory board or a dual port facility for shared memory. The Multibus II Message Passing Facility is intended to simplify the design and configuration of such systems while maintaining an equivalent performance of the more tightly coupled design (Figure 1-4).



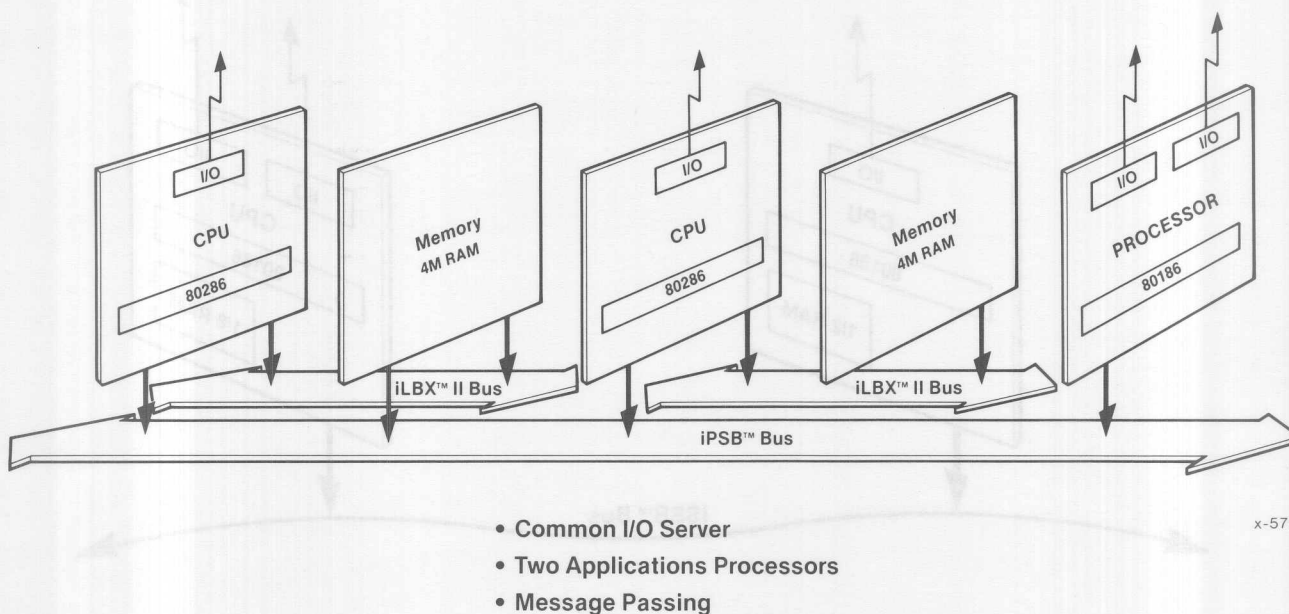


Figure 1-4. Enhanced Support of Multiple Processors

### 1.3.3 Lower Cost Requirements

With an successful system, competition will attempt to undermine the leader's market share with lower cost. Furthermore, a low-cost system design is often required to achieve design wins that lead to future product sales of more costly systems. The Multibus II Serial System Bus allows a low-cost implementation of the message-passing facility of the Parallel System Bus with complete software compatibility. Thus, the software investment is protected and leveraged in low-cost designs based on highly integrated processors such as the 80186 (Figure 1-5).

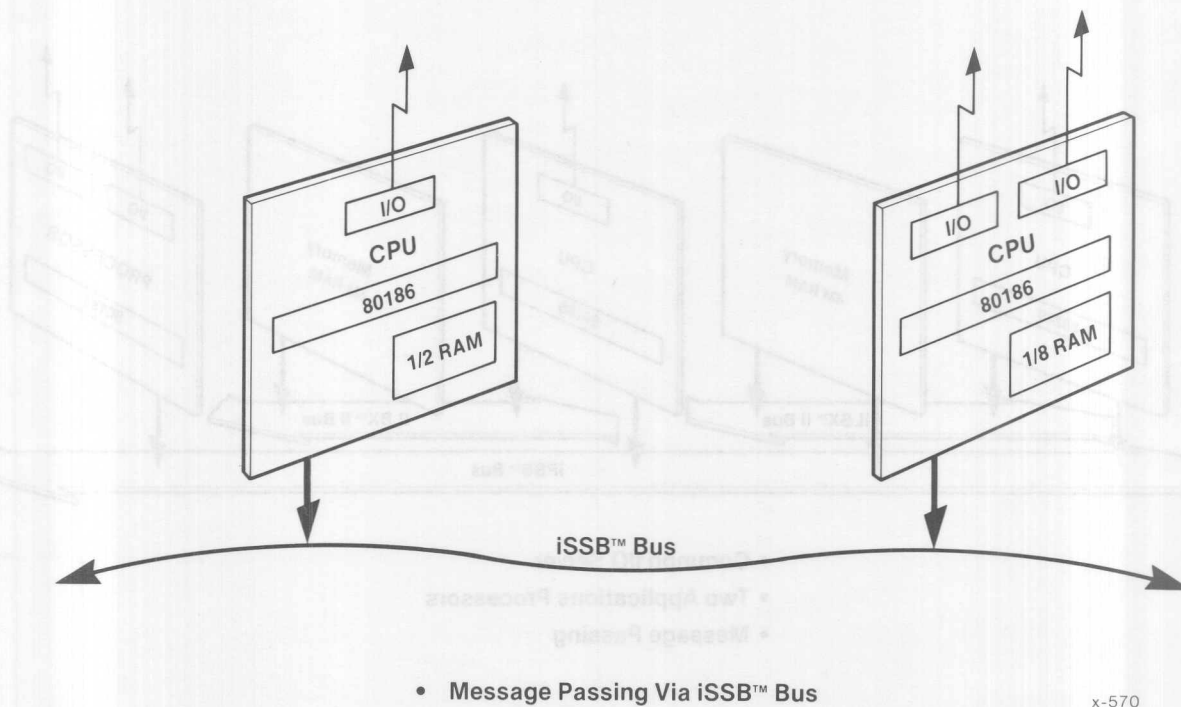


Figure 1-5. Low-Cost System Design

#### 1.4 SUMMARY

This overview has attempted to present the characteristics of the Multibus II bus architecture, an advanced open system multiple bus architecture. Its intent has also been to demonstrate the attributes of the Multibus II busses: the 32-bit Parallel System Bus with its 40 megabytes/sec throughput and effective support of multiple processors; the iLBX II bus with its high-speed access to large amounts of off-board memory; and the Serial System Bus, a low-cost alternative to the message-passing facilities of the Parallel System Bus.

Individually, the busses represent significant advances in bus design. Together, they represent an evolutionary path to future VLSI technology. The Multibus II specification which follows defines the standard protocol, electrical, and mechanical requirements for each bus structure in Chapters 1-5. In Chapter 6, the specification also defines the common system interface which allows the busses to be combined for a total system architecture.

\*\*\*



# INTEL iPSB™ BUS SPECIFICATION

---







# CONTENTS

	PAGE
PARALLEL SYSTEM BUS SPECIFICATION	
1.1 Scope.....	2-1
1.2 Object.....	2-1
1.3 Definitions.....	2-2
1.4 MULTIBUS™ II Architecture Overview.....	2-4
1.5 Parallel System Bus Overview.....	2-5
1.5.1 Address/Data Path.....	2-6
1.5.2 Message Passing Facilities.....	2-6
1.5.3 Interconnect Facility.....	2-6
1.5.4 Synchronous Operation of the Parallel System Bus.....	2-7
1.5.5 Bus Cycles on the Parallel System Bus.....	2-7
2. Signal Descriptions.....	2-13
2.1 General.....	2-13
2.2 Signal Naming and Notational Conventions.....	2-13
2.3 Signal Groups.....	2-15
2.3.1 Arbitration Cycle Signal Group.....	2-15
2.3.2 Address/Data Bus Signal Group.....	2-16
2.3.3 System Control Signal Group.....	2-17
2.3.4 Exception Cycle Signal Group.....	2-23
2.3.5 Central Control Signal Group.....	2-24
2.3.6 Power.....	2-25
3. Parallel System Bus Protocol.....	2-26
3.1 General.....	2-26
3.2 Arbitration Cycle Overview.....	2-27
3.2.1 Bus Arbitration Priority.....	2-28
3.2.2 Arbitration Priority Example.....	2-29
3.2.3 Arbitration ID and Cardslot ID Assignment.....	2-31
3.2.4 Arbitration Sequence.....	2-34
3.3 Transfer Cycle Overview.....	2-38
3.3.1 Types of Transfer Cycles.....	2-38
3.3.3 Transfer Cycle LOCK Operation.....	2-45
3.3.4 Errors In Transfer Cycles.....	2-45
3.3.5 Address Space Definitions in Transfer Cycles.....	2-49
3.3.6 Data Width During Transfer Cycles.....	2-54
3.3.7 Data Alignment Error Reporting During Transfer Cycles.....	2-58
3.3.8 Data Alignment Interface Example.....	2-59
3.3.9 Interrupts.....	2-59
3.4 Exception Cycle Overview.....	2-62
3.4.1 Causes of Exception Cycles.....	2-63
3.5 Central Control Functions.....	2-65
3.5.1 Power-Up Function (Cold-Start).....	2-66
3.5.2 Initialization Sequence (Warm-Start).....	2-67
3.5.3 Power Failure And Recovery Sequence.....	2-68



# CONTENTS (continued)

	PAGE
PARALLEL SYSTEM BUS SPECIFICATION (continued)	
3.6 State-Flow Diagrams.....	2-69
3.6.1 Notation in the State-Flow Diagrams.....	2-70
3.6.2 State-Flow Sequence for an Agent Monitoring the Bus.....	2-70
3.6.3 State-Flow Sequence for an Arbitration Cycle.....	2-72
3.6.4 State-Flow Sequence for Requesting Agents in a Transfer Cycle..	2-73
3.6.5 State-Flow Diagram for Replying Agents in a Transfer Cycle..	2-77
3.6.6 State-Flow Diagrams for Exception Cycles.....	2-80
3.7 Central Services Module.....	2-80
4. Electrical Characteristics.....	2-81
4.1 General.....	2-81
4.2 AC Timing Specifications.....	2-81
4.3 DC Specifications for Signals.....	2-88
4.4 Current Limitations Per Connector.....	2-91
4.5 Pin Assignments.....	2-92
5. Compliance Levels.....	2-93
5.1 Introduction.....	2-93
5.2 Data Path.....	2-93
5.3 Address Path.....	2-93
5.4 Compliance Codes.....	2-93
5.5 Compliance Statement Example.....	2-94

## TABLES

2-1. Signal Notation Summary.....	2-13
2-2. Signal Level Requirements.....	2-14
2-3. Signal Groups on the Parallel System Bus.....	2-15
2-4. Summary Of Functions Of SC9* through SC0*.....	2-20
2-5. Functions Of SC9*-SC0* During REQUEST Phase.....	2-21
2-6. Functions Of SC9*-SC0* During REPLY Phase.....	2-22
3-1. Arbitration Priority Protocol.....	2-29
3-2. Default Power-Up ID Assignment.....	2-31
3-3. Arbitration ID Comparison Example.....	2-34
3-4. Address Space Summary.....	2-50
3-5. Data Alignments for Operations Using Memory, I/O and Interconnect Address Space.....	2-55
3-6. Data Alignment for Message Space Operations.....	2-56
3-7. Data Alignment and Width Errors for Memory, I/O and Interconnect Space Operations.....	2-58
3-8. Data Alignment and Width Errors for Message Space.....	2-59





## CONTENTS (continued)

### TABLES (continued)

	PAGE
PARALLEL SYSTEM BUS SPECIFICATION (continued)	
4-1. Clock Specification.....	2-81
4-2. Timing Parameters For Signal Driver.....	2-83
4-3. Timing Parameters For Signal Receivers.....	2-84
4-4. Cold-Start Control Timing.....	2-85
4-5. Warm-Start Control Timing.....	2-86
4-6. Power Failure and Recovery Timing.....	2-87
4-7. DC Specifications For Signal Drivers.....	2-88
4-8. DC Specifications For Signal Receivers.....	2-89
4-9. Backplane Termination Requirements.....	2-90
4-10. Power Limitations for a One-Connector Agent.....	2-91
4-11. Power Limitations for a Two-Connector Agent.....	2-91
4-12. Parallel System Bus Connector Pinout.....	2-92

### FIGURES

1-1. MULTIBUS™ II Bus Architecture .....	2-4
1-2. Block Diagram Of The Parallel System Bus Interface.....	2-5
1-3. Bus Cycle Relationships.....	2-8
1-4. Block Diagram of Bus Cycle Operation.....	2-10
1-5. Block Diagram of Bus Cycle Operation with Errors.....	2-12
3-1. Bus Cycle Details.....	2-27
3-2. Typical Arbitration Cycle Sequence.....	2-30
3-3. Arbitration ID Assignment Timing Sequence.....	2-32
3-4. Arbitration ID Interface Example (At Each Agent).....	2-33
3-5. Timing Sequence for Bus Control Acquisition.....	2-35
3-6. Timing Sequence With Bus Locked.....	2-36
3-7. Timing Sequence for Bus Release.....	2-37
3-8. Block Diagram of Single-Transfer READ Operation.....	2-39
3-9. Timing For Single-Transfer READ Operation.....	2-40
3-10. Block Diagram of Single Transfer WRITE Operation.....	2-41
3-11. Handshake In Single-Transfer WRITE Operation.....	2-41
3-12. Block Diagram of Sequential-Transfer WRITE Operation.....	2-42
3-13. Timing For Sequential-Transfer WRITE Operation.....	2-43
3-14. Block Diagram Of Broadcast Operation.....	2-44
3-15. Timing For Broadcast Operation.....	2-44
3-16. Timing Sequence For LOCK* Signal Operation.....	2-45
3-17. Timing Sequence for Transfer Width Agent Error.....	2-48
3-18. Timing Sequence for Continuation Agent Error.....	2-49
3-19. Address Format For Operations Using Memory Address Space...	2-51
3-20. Block Diagram Of Sequential Data Transfers In Memory Space.	2-51
3-21. Address Format For Operations Using I/O Address Space.....	2-52
3-22. Address Format For Operations Using Message Space.....	2-53



## CONTENTS (continued)

### FIGURES (continued)

	PAGE
PARALLEL SYSTEM BUS SPECIFICATION (continued)	
3-23. Address Format For Operations Using Interconnect Space.....	2-54
3-24. 16-Bit and 32-Bit Message Format Examples.....	2-57
3-25. Interface Requirements For Data Alignment.....	2-60
3-26. Interrupt Message Format.....	2-61
3-27. Interrupt Message Sequence.....	2-62
3-28. Exception Cycle Signal Relationships.....	2-63
3-29. Effects of a TIMEOUT* Exception on Bus Cycles.....	2-64
3-30. Effects of a BUSERR* Exception on Bus Cycles.....	2-65
3-31. Power-On System Reset Sequence.....	2-66
3-32. Warm-Start Reset Sequence.....	2-67
3-33. Power Failure Recovery Sequence.....	2-69
3-34. State-Flow Diagram For Requesting Agents Monitoring the Bus	2-71
3-35. State-Flow Diagram for a Requesting Agent in an Arbitration Cycle.....	2-74
3-36. State-Flow Diagram for Requesting Agents in a Transfer Cycle.....	2-76
3-37. State-Flow Diagram for Replying Agents in a Transfer Cycle.	2-79
4-1. BCLK* and CCLK* Timing Relationship.....	2-82
4-2. Driver Timing Parameters.....	2-82
4-3. Receiver Timing Parameters.....	2-83
4-4. Cold-Start Timing Parameters.....	2-85
4-5. Warm-Start Timing Parameters.....	2-86
4-6. Power Failure and Recovery Timing Parameters.....	2-87



## CHAPTER 2

# PARALLEL BUS SPECIFICATION

### 1.1 SCOPE

This specification gives a description of the operation, functions, and attributes of the Parallel System (iPSB) bus portion of the Multibus II bus architecture. The Parallel System bus provides the facilities for parallel data transfers and for those system functions that must be shared by all parts of a Multibus II bus architecture.

The specification applies to microprocessor computer systems or portions of them where:

- 1) the data transmission rate does not exceed 40 megabytes per second, the maximum sustained bandwidth.
- 2) the length of the data path does not exceed 16.8 inches.
- 3) the data exchanged among boards is digital (rather than analog).
- 4) the total number of boards connected to one contiguous bus does not exceed 20.

### 1.2 OBJECT

This standard is intended:

- 1) To define a general-purpose parallel interface for use with microcomputer equipment.
- 2) To specify the electrical and functional interface requirements that each connected unit shall meet in order to be interconnected to and communicate on this bus.
- 3) To specify terminology and definitions related to the Parallel System bus.
- 4) To guide independent manufacturers in designing computer equipment that is architecturally compatible.



## 1.3 DEFINITIONS

The following definitions apply for the iPSB Bus Specification. This section contains only general definitions; more specific definitions are provide in other sections as appropriate.

Agent	A physical unit which has an interface to the Parallel System bus.
Module	A basic functional unit within an agent. An agent may be comprised of one or more modules, each performing a specific function.
Central Services Module (CSM)	A specific module that is required in all systems using the Multibus II bus architecture. The CSM provides services required by all agents on the Parallel System bus, such as starting certain bus cycles and guaranteeing uniform initialization of all agents. The CSM is always located in a specific slot in the system backplane. The Parallel System bus supports operation with one and only one CSM per system.
Interface	A shared boundary between modules or agents of a computer system, through which information is conveyed.
Bus Cycle	The basic unit of processing whereby digital signals effect the transfer of data across an interface by means of a sequence of control signals and an integral number of clock cycles. The Parallel System bus allows three types of bus cycles, all measured in terms of integral numbers of clock cycles. Each type of bus cycle is initiated with the request and concluded with the reply. The cycles on the Parallel System bus are the arbitration cycle, the transfer cycle, and the exception cycle.
Arbitration Cycle	The bus cycle in which agents attempt to gain exclusive access to the Parallel System bus. The length of the arbitration cycle depends on the number of agents requesting access to the bus. For each agent that requests access, the arbitration cycle includes two phases, the resolution phase and acquisition phase.
Resolution Phase	The initial phase of an arbitration cycle in which all agents requesting access to the bus drive an arbitration ID onto the Parallel System bus. Agents mutually resolve requests and allow the agent with the highest priority to gain access to the bus.

## PARALLEL SYSTEM BUS SPECIFICATION

**Acquisition Phase** One agent at a time (the one given ownership of the bus) enters the acquisition phase of the arbitration cycle after determining that it has the highest priority. The agent in the acquisition phase is referred to as the bus owner and immediately begins conducting transfer cycles.

**Transfer Cycle** The bus cycle in which a bus owner transfers data on the Parallel System bus. The transfer cycle is subdivided in two phases, the request phase and the reply phase.

**Request Phase** The initial phase of a transfer cycle in which the bus owner requests a data transfer operation. The bus owner places command and address information on the Parallel System bus.

**Reply Phase** The final phase of a transfer cycle in which another agent replies to the data transfer request from the bus owner. The phase consists of one or more consecutive data and/or status transfers on the Parallel System bus.

**Exception Cycle** The bus cycle in which an agent places an error indication onto the Parallel System Bus and terminates all bus cycles. The exception cycle can be subdivided into two phases, a signal phase and a recovery phase.

**Signal Phase** The initial phase of an exception cycle in which an agent on the bus places an exception error indication on the Parallel System bus and terminates all arbitration and transfer cycles. During this phase all agents are notified of the error condition.

**Recovery Phase** The final phase of an exception cycle in which the Parallel System bus is allowed to sit idle for a defined amount of time. The idle time allows the condition of the bus lines to stabilize before more bus cycles begin.

**Requesting Agent** The agent that initiates the arbitration cycle and transfer cycles. The requesting agent places a request for a specific operation onto the Parallel System bus.

**Replying agent** The agent or agents with which the requesting agent performs a transfer cycle. Replying agents respond to a request from a requesting agent during the transfer cycle.

**Exception** An abnormal condition on the bus caused by either a parity error or a bus timeout.

**Agent Error** An error condition in agent caused by improper data width, no more memory, or agent busy during message.

# PARALLEL SYSTEM BUS SPECIFICATION

## 1.4 MULTIBUS® II ARCHITECTURE OVERVIEW

Figure 1-1 shows the Multibus II bus architecture and how the Parallel System bus contributes to that architecture.

As Figure 1-1 shows, the Multibus II bus architecture provides three separate busses: the Parallel System Bus (iPSB bus), the Local Execution Bus (iLBX bus), and the Serial System Bus (iSSB bus). In some cases the functions of busses overlap, however, each bus is optimized to add a particular attribute to the Multibus II architecture.

The three busses provide you with the ability to put together the system environment that best suits the cost/performance and bandwidth/latency goals that you require in your application.

The iPSB bus is intended to be a general purpose interface for multiple requesting agents. Figure 1-2 shows a functional diagram of the Parallel System bus. As the figure shows, the Parallel System bus consists of five groups of signals that requesting and replying agents use to communicate with one another. The iPSB bus can be extended to a maximum of 20<sup>+</sup> agents and 16.8 inches. Each system includes a Central Services Module (CSM) that coordinates the services common to all agents in the system.

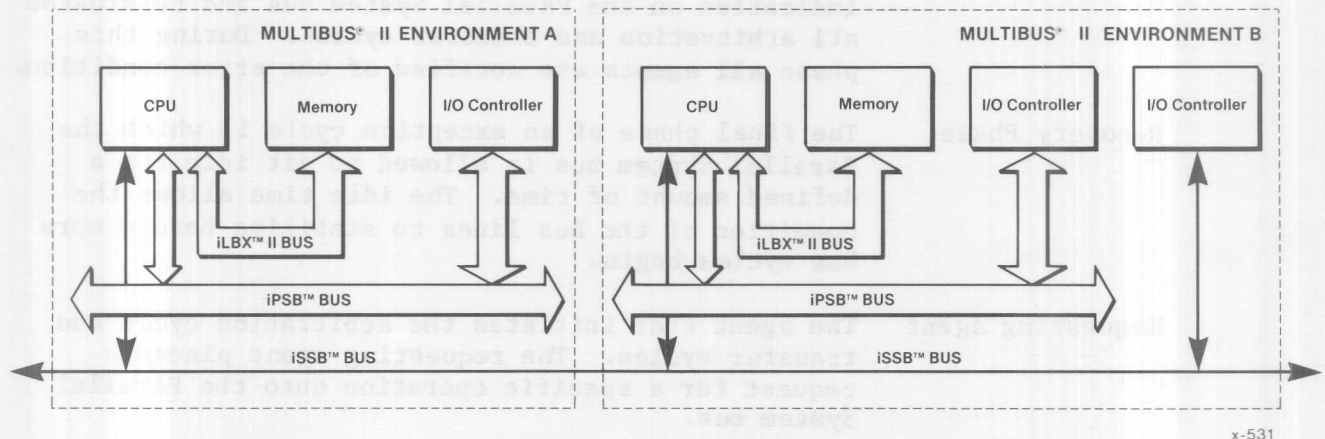


Figure 1-1. MULTIBUS® II Bus Architecture

## PARALLEL SYSTEM BUS SPECIFICATION

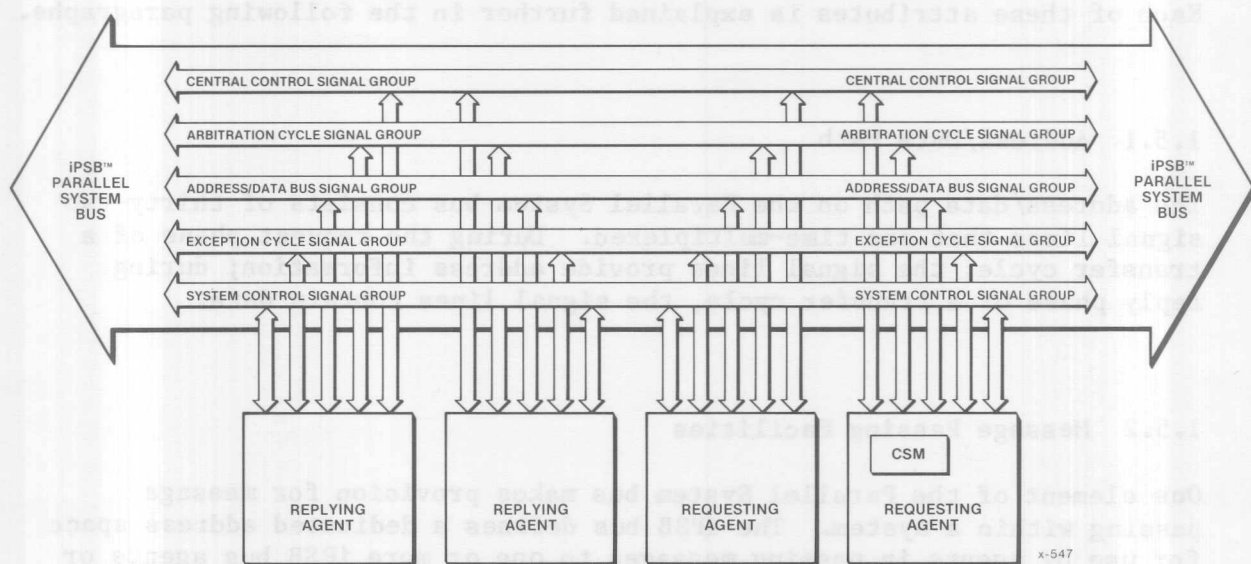


Figure 1-2. Block Diagram Of The Parallel System Bus Interface

### 1.5 PARALLEL SYSTEM BUS OVERVIEW

The Parallel System bus has several specific attributes that make it a unique part of the Multibus II bus architecture, as follows:

- 1) the address/data path is 32-bits wide, providing a 4-gigabyte address range and capable of 8-, 16-, 24-, or 32-bit transfers.
- 2) the message-passing facility supports inter-agent and inter-bus communication in the architecture.
- 3) the interconnect facility allows configuration of agents and modules on the Parallel System bus.
- 4) the bus operations are synchronous; the bus uses a handshaking protocol that is synchronous with the basic clock rate (10 megahertz) for the system.



## PARALLEL SYSTEM BUS SPECIFICATION

- 5) the bus operates in defined bus cycles. The three types of bus cycles are: arbitration cycle, transfer cycle, and exception cycle.
- 6) the bus ensures uniform system operation; the bus uses a Central Services Module to provide some centralized system functions.

Each of these attributes is explained further in the following paragraphs.

### 1.5.1 Address/Data Path

The address/data path on the Parallel System bus consists of thirty-two signal lines that are time-multiplexed. During the request phase of a transfer cycle, the signal lines provide address information; during reply phase of a transfer cycle, the signal lines contain data.

### 1.5.2 Message Passing Facilities

One element of the Parallel System bus makes provision for message passing within a system. The iPSB bus defines a dedicated address space for use by agents in passing messages to one or more iPSB bus agents or to bus agents on the iSSB bus in the Multibus II bus architecture.

The message passing facility provides a standardized method for performing direct transfers of messages (data) from one agent to another.

Message passing on the iPSB bus provides for inter-agent communications (interrupts) and data movement.

### 1.5.3 Interconnect Facility

The interconnect facility within the Parallel System bus makes provision for dynamic, software-controlled, initialization and identification of an agent in a system. The iPSB bus defines a dedicated address space that contains operating specifications for each agent on the iPSB bus interface.

#### 1.5.4 Synchronous Operation of the Parallel System Bus

The Parallel System bus is referred to as being "synchronous" in that all events on the Parallel System bus occur relative to the active edge of a bus clock that is distributed to all active agents on the bus. The synchronous nature of the bus does not place rigid time constraints on the length of a transfer cycle. Rather, it requires that agents sample all signals at a specific time with respect to the bus clock.

The synchronous nature of the Parallel System bus allows defining of all operations on the bus interface as a sequence of bus states that are diagrammed later in this text. Each bus state correlates to a control sequence provided by the agents on the bus. In some cases, the bus states allow a specific signal on the bus to provide a different meaning or function depending on the state of the bus.

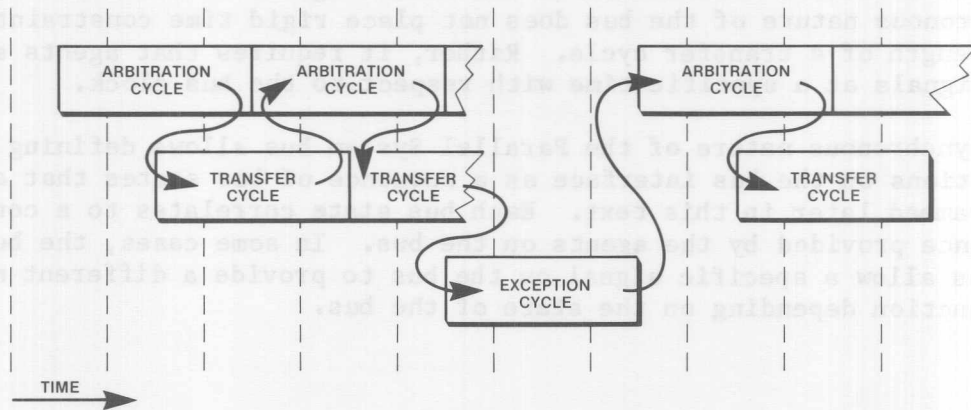
#### 1.5.5 Bus Cycles On The Parallel System Bus

Agents perform one of three types of bus cycles on the Parallel System bus: an arbitration cycle, a transfer cycle, or an exception cycle. Each cycle has a definite event that signals the start and end of the cycle.

A typical read or write operation on the Parallel System bus consists of a series of three cycles. First, the requesting agent arbitrates for access to the bus. When it gains control, the agent performs one or more transfer cycles to read or write data. If the replying agent senses an exception, the exception indication initiates an exception cycle to terminate the transfer cycle.

Figure 1-3 shows how the three types of cycles form a typical read or write operation on the bus. The arbitration cycle is a time-period in which agents decide which will be the next to control the bus; the transfer cycle is a subsequent time-period in which that agent performs the addressing and data transferring portions of the operation; and the exception cycle is an error reporting time-period that occurs only when an error is sensed.

Each of the three types of bus cycle is described further in the following paragraphs.



x-544

Figure 1-3. Bus Cycle Relationships

**1.5.5.1 ARBITRATION CYCLE.** The arbitration cycle is a defined time-period during which all agents that require access to the bus collectively arbitrate for access rights to the bus. The beginning of the arbitration cycle is the result of access requests from agents and the end of the arbitration cycle occurs when the last agent requesting access to the bus at the beginning of the cycle obtains access to the bus.

A typical arbitration cycle consists of several resolution/acquisition phase sequences. However, during each sequence, only one agent is allowed access to the bus at a time.

Figure 1-4 shows a more detailed diagram that includes three, consecutive, errorless operations on the bus. This type of operation sequence is typical of a system with more than one requesting agent. The first operation passes one data, the second passes three data, and the third passes one data. The diagram shows the relationship between the arbitration cycle and the transfer cycle and shows the two phases of each arbitration cycle: the resolution phase and the acquisition phase.

The resolution phase is the time-period in which all agents collectively arbitrate for access rights to the bus. The agents decide among themselves which of them is going to control the bus after the current transfer cycle.

## PARALLEL SYSTEM BUS SPECIFICATION

The arbitration method for the Parallel System bus is referred to as a parallel contention or self-selecting method. In this method, each agent on the bus is assigned an encoded bit pattern that gives it a priority level when it makes a request for bus use.

The agent that has the highest priority request during the resolution phase obtains access rights to the bus and begins the acquisition phase of the arbitration cycle. As such, the agent becomes what this specification refers to as the bus owner.

Once in the acquisition phase, the agent begins its transfer cycle and causes the other agents to hold the arbitration logic in a resolution phase (resolving for next access rights) until the transfer cycle is completed. Figure 1-4 shows the end of a transfer cycle (EOC) initiating an acquisition phase of the arbitration cycle to allow the next bus owner to acquire control of the bus.

1.5.5.2 TRANSFER CYCLE. After arbitrating for and winning control of the bus, an agent performs transfer cycles to move data to or from another agent.

Figure 1-4 shows a more detailed diagram that includes three consecutive transfer cycles on the bus. The diagram shows the major component parts of the transfer cycle, the request phase and the reply phase, and shows the active components of each phase, including the command, the address, the data, the handshake and the EOC.

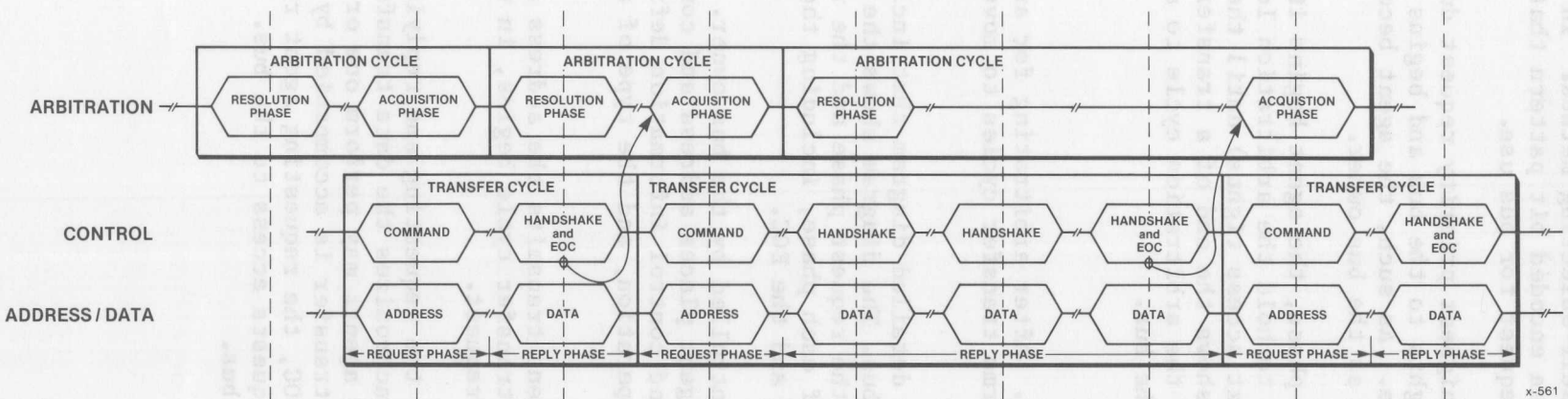
The request phase is controlled by the bus owner. During the request phase, the requesting agent places address and control information onto the bus. The address and control information defines the replying agent(s), the type of operation, and the type of address space involved in the transfer cycle.

After the requesting agent transmits the address and control information, the reply phase of the transfer cycle begins, in which the replying agent(s) satisfies the request.

During the reply phase, the requesting and replying agents engage in a close handshake that synchronizes the data transfer sequence. The requesting and replying agents may perform one or more cycles in a reply phase. The final data transfer is accompanied by an end-of-cycle (EOC) indication. With the EOC, the requesting agent releases ownership of the bus if another agent requests access to the bus. Otherwise, the agent keeps ownership of the bus.



# PARALLEL SYSTEM BUS SPECIFICATION



x-561

Figure 1-4. Block Diagram Of Bus Cycle Operation

## PARALLEL SYSTEM BUS SPECIFICATION

1.5.5.3 EXCEPTION CYCLE. If a replying agent detects an exception during a transfer cycle, the agent provides an exception indication on the bus that causes the requesting agent to begin an exception cycle. The exception cycle terminates both arbitration cycles and transfer cycles.

Figure 1-5 shows the same operation sequence presented in Figure 1-4, except that in Figure 1-5 the replying agent senses an exception that initiates an exception cycle. The diagram shows the two phases of the exception cycle, the signal phase and the recovery phase, and shows how the exception cycle terminates transfer and arbitration cycles.

During the signal phase, the agent that detected the exception condition presents an indication of the problem to all agents on the bus via exception lines. As a result of the signal phase, all agents terminate any arbitration cycles and transfer cycles that are in progress.

After all agents terminate their arbitration and transfer cycles, the recovery phase of the exception cycle begins. During the recovery phase, the bus is idle for a fixed amount of time. This allows the bus a fixed amount of idle-time before resuming operation. When the fixed delay of the recovery phase expires, the exception cycle ends. At that point, agents may resume arbitration and transfer cycles.

As Figure 1-5 shows, the exception cycle has a specific time relationship with respect to sensing an exception in the transfer cycle. That is, the current exception cycle reflects an exception condition sensed during the previous transfer cycle. This time relationship defines a time period in which system exceptions are signalled on the bus.

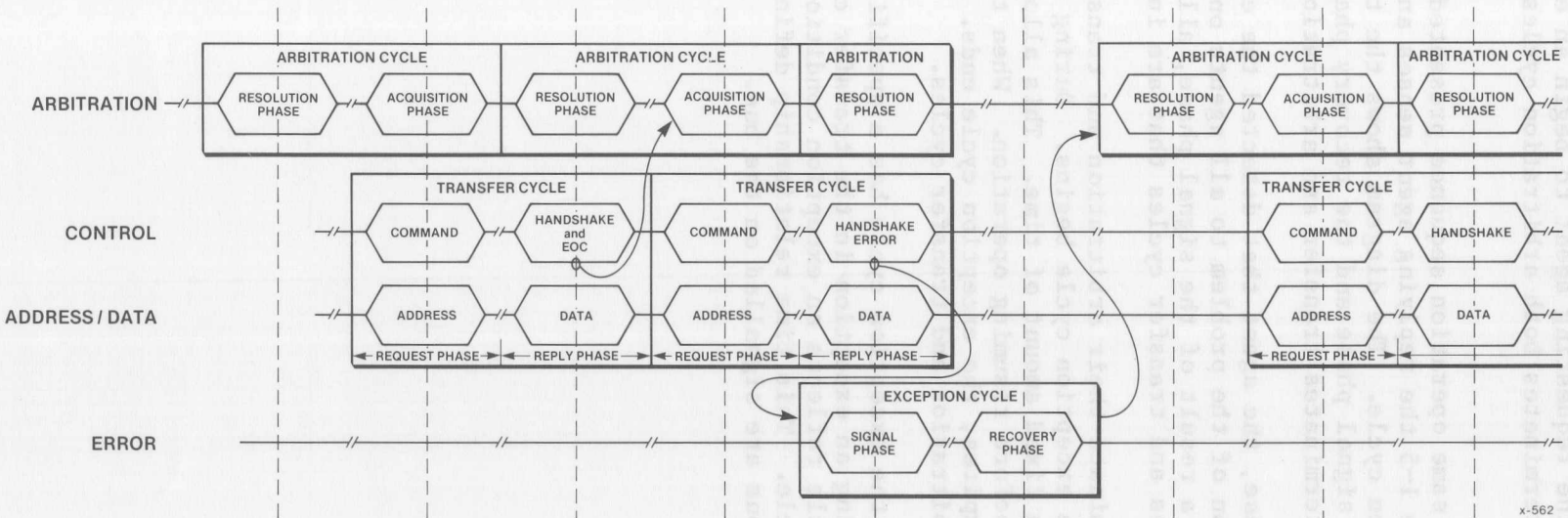


Figure 1-5. Block Diagram of Bus Cycle Operation with Errors

# PARALLEL SYSTEM BUS SPECIFICATION

## 2. SIGNAL DESCRIPTIONS

### 2.1 GENERAL

This section of the specification lists the signal groups, names each signal on the Parallel System bus, and describes the functions of each signal. The signals on the Parallel System bus are presented in five groups: the arbitration cycle signal group, the transfer cycle signal group, the address/data bus signal group, the exception cycle signal group, and the system control signal group.

### 2.2 SIGNAL NAMING AND NOTATIONAL CONVENTIONS

This section of the text, as well as throughout the specification, uses a clear and consistent notation for naming signals. The terms one:zero and true:false can be ambiguous, so their use is avoided. In their place, the terms electrical high and low (H and L) are used. An asterisk following a signal name indicates that the signal is active when low.

Table 2-1 further explains the signal naming notation used in this specification.

Table 2-1. Signal Notation Summary

Signal Name	Electrical Notation	Logical Notation	State
BREQ	H	1 or True	Active
	L	0 or False	Inactive
BREQ*	L	1 or True	Active
	H	0 or False	Inactive



# PARALLEL SYSTEM BUS SPECIFICATION

Many signals on the Parallel System bus are more easily or conveniently discussed as a group. Names for these signals follow a decimal radix numbering convention. Within each signal group, the least significant bit of the group has the suffix '0' following the group name. Successively higher order bits are given higher decimal number suffixes. As an example, AD31\* through AD0\* refers to the 32 address/data signal lines.

Some of the Parallel System bus signals have a lower case 'n' following the signal name. All signals whose name follows this convention, for example LACHn\*, are connected to only one cardslot (numbered 'n') in the system cardcage.

The signal level requirements for the iPSB bus signals are listed in Table 2-2.

Table 2-2. Signal Level Requirements

0.0V <	Driver Low Output Level = Vol <	0.55V at max.	Iol
2.4V at max.	Ioh <	Driver High Output Level = Voh <	5.0V
-0.5V <	Receiver Low Output Level = Vil <	0.8V	
2.0V <	Receiver High Output Level = Vih <	5.25V	

State	Logical Notation	Electrical Notation	Signal Name
Active	1 or True	H	REQ
Inactive	0 or False	L	
Active	1 or True	H	REQ*
Inactive	0 or False	L	

## PARALLEL SYSTEM BUS SPECIFICATION

### 2.3 SIGNAL GROUPS

The Parallel System bus contains five groups of signals over which requesting and replying agents can enact the protocol. The signal groups and their functions are listed in Table 2-3.

Table 2-3. Signal Groups On the Parallel System Bus

Signal Group Name	Description
Arbitration Cycle Signal Group	Provides the bus requesting and bus granting signals and identifies the priority of a bus request.
Address/Data Bus Signal Group	Provides the address, data, and parity signals for read and write operations.
System Control Signal Group	Provides the control signals needed to transfer address and data on the address/data signal group.
Exception Cycle Signal Group	Provides the error detection signals to stop transfer cycles.
Central Control Signal Group	Provides system-wide services such as reset and initialization control.

#### 2.3.1 Arbitration Cycle Signal Group

The arbitration signals on the Parallel System bus determine which agent gains exclusive access to the Parallel System bus (which agent is the bus owner). All requesting agents that require access to bus resources must arbitrate for use of the bus. On being granted bus ownership, an agent begins using the address/data lines to perform a transfer cycle.

The seven signals within the arbitration cycle signal group that implement the arbitration cycle protocol on the Parallel System bus are: bus request (BREQ\*) signal and the arbitration ID signals (ARB5\*-ARB0\*). Each is described in the following paragraphs.

**2.3.1.1 BREQ\* (Bus Request).** The bus request signal is called BREQ\*. All agents that require access to the bus assert the BREQ\* signal. All agents that activate the BREQ\* signal in the resolution phase of the arbitration cycle enter into arbitration for access to the bus. The BREQ\* signal is an OR-tied signal that is bussed on the backplane; all agents in the system share the same bus request line.

## PARALLEL SYSTEM BUS SPECIFICATION

2.3.1.2 ARB5\* THROUGH ARB0\* (Arbitration). The arbitration signals are called ARB5\* through ARB0\*. The arbitration signals provide three functions:

- o a cardslot ID
- o an arbitration ID
- o arbitration

On power-up initialization, the CSM uses the ARB5\* through ARB0\* lines to assign the agent in each cardslot a unique cardslot ID on power-up. The CSM drives the cardslot ID onto the ARB4\* through ARB0\* lines whenever the ARB5\* signal is low. The cardslot ID gives the agent a geographical address and provides the cardslot ID used in addressing interconnect space. At system initialization, all agents are expected to latch the cardslot ID provided by the CSM on the ARB4\* through ARB0\* lines.

On power-up initialization, the CSM uses the ARB5\* through ARB0\* lines to assign each agent a unique arbitration ID on power-up. The CSM drives the arbitration ID onto the ARB4\* through ARB0\* lines whenever the ARB5\* signal is high. The arbitration ID (ARB4\* through ARB0\*) is a unique identification number provided to each agent by the CSM during system initialization. The identification number provides a means for each agent to enter an arbitration cycle and provides an arbitration priority for each agent. At system initialization, all agents are expected to latch an identification number provided by the CSM on the ARB4\* through ARB0\* lines.

After power-up initialization, each agent drives its identification number (the same one that it received from the CSM) back onto ARB5\* through ARB0\* only when it requires access to the bus. The ARB5\* through ARB0\* signals are OR-tied signals that makes it possible for agents to place their arbitration ID onto the bus (request access to the bus) at the same time. The high priority request signal, ARB5\*, is an OR-tied signal line that allows an agent to make a high priority request. By holding ARB5\* active, an agent gains priority over other agents not asserting ARB5\*. If more than one agent asserts ARB5\* during an arbitration cycle, then bus owner is the agent with the higher priority within the ARB4\* through ARB0\* lines.

### 2.3.2 Address/Data Bus Signal Group

Only the requesting agent that is the bus owner and the selected replying agent(s) use the address/data signals on the Parallel System bus. The address/data bus signal group includes two sets of signals: address/data signals and parity signals for each byte of address/data. Each set of signals is described in the following paragraphs.

## PARALLEL SYSTEM BUS SPECIFICATION

**2.3.2.1 AD31\* THROUGH ADO\* (Address/Data Bus).** The address/data signals on the Parallel System bus are AD31\* through ADO\*. These 32 signals are multiplexed and serve a dual purpose depending on the phase of the transfer cycle.

During the request phase of the transfer, they contain the address for the ensuing transfer cycle (ADO\* is the least significant and AD31\* is the most significant address bit). This address refers to a location for memory or I/O spaces, a processing agent in message space and a board or slot location in interconnect space. The requesting agent drives these lines during the request phase.

During the reply phase of the transfer, they contain either eight, sixteen, twenty-four, or thirty-two bits of data. Again, ADO\* is the least significant and AD31\* is the most significant data bit. The replying agent drives the AD31\* through ADO\* lines during the reply phase of a READ operation.

**2.3.2.2 PAR3\* THROUGH PAR0\* (Parity).** The parity signals on the Parallel System bus are PAR3\* through PAR0\*. The parity signals ensure the integrity of data transferred on AD31\* through ADO\*. Each parity signal must generate even parity for one byte of a 4-byte (32-bit) address/data bus. An agent that drives less than 4 bytes of data need not place parity for the missing bytes onto the bus. The parity signals are assigned to bytes as follows:

PAR0*	parity for AD7* through ADO*
PAR1*	parity for AD15* through AD8*
PAR2*	parity for AD23* through AD16*
PAR3*	parity for AD31* through AD24*

Even parity means that when the number of low bits on the bus is even, the corresponding parity line is high. If the number is odd, the parity line is low.

The agent that drives AD31\* through ADO\* is responsible for generating parity on the bus. All agents check parity, both during the request phase and during the handshake(s) in the reply phase.

### 2.3.3 System Control Signal Group

The system control signal group on the Parallel System bus consists of a set of ten signals, SC9\* through SC0\*, that provide control between agents during transfer cycles. Agents use SC9\* through SC0\* to define commands or report status, depending on the phase of the transfer cycle.

During the request phase of a transfer cycle, the requesting agent drives SC9\* through SC0\*. The SC lines provide command information to the replying agent(s).



## PARALLEL SYSTEM BUS SPECIFICATION

During the reply phase of a transfer cycle, the requesting agent drives the SC9\*, SC3\*, SC2\*, SC1\*, and SC0\* signals and the replying agent drives the SC8\*, SC7\*, SC6\*, SC5\*, and SC4\* signals. The SC lines provide handshake and status facilities between requesting and replying agents.

The following paragraphs describe the functions of each SC line during request phases and reply phases. Table 2-4 summarizes the functions of the SC lines. Tables 2-5 and 2-6 list the request phase and reply phase functions for each of the SC lines.

**2.3.3.1 SC0\* (Request Phase).** The SC0\* line is always driven by the requesting agent and does the same function for request and reply phases of a transfer cycle. When low, SC0\* indicates to all agents that the requesting agent is in the request phase of a transfer cycle and that the address/data bus and the SC lines contain valid request phase information. When high, SC0\* indicates that there is no request phase.

**2.3.3.2 SC1\* (Lock).** The SC1\* line is always driven by the requesting agent and does the same function during request and reply phases of a transfer cycle. When a requesting agent holds SC1\* low, the agent locks any multi-ported resources with which it is communicating. In doing so, the requesting agent guarantees itself exclusive access to a multi-ported bus resource and retains bus ownership for more than one transfer cycle. The requesting agent locks the bus and resources by driving SC1\* low and unlocks them by driving SC1\* high.

**2.3.3.3 SC2\* (Data Width Or End-Of-Cycle).** The SC2\* line changes functions depending on the phase of the transfer cycle.

During the request phase, the requesting agent uses SC2\* (with SC3\*) to tell the replying agent the data-width format for the ensuing transfer. The requesting agent codes SC2\* and SC3\* into four options: 8-, 16-, 24-, or 32-bit data-width.

During the reply phase, the requesting agent uses SC2\* to inform all agents that the current data transfer is the last, the end-of-cycle.

**2.3.3.4 SC3\* (Data Width Or Requestor Ready).** The SC3\* line changes functions depending on the phase of the transfer cycle.

During the request phase, the requesting agent uses SC3\* signal, along with SC2\*, to tell the replying agent what the data-width is for the ensuing transfer. The requesting agent has four options: 8-, 16-, 24-, or 32-bit data-width.

## PARALLEL SYSTEM BUS SPECIFICATION

During the reply phase, the requesting agent holds SC3\* low to notify the replying agent when it is ready to send or receive data. SC3\* works closely with SC4\* to provide a two-directional handshake during the reply phase of the transfer cycle.

**2.3.3.5 SC4\* (Address Space Or Replier Ready).** The SC4\* line changes functions depending on the phase of the transfer cycle.

During the request phase, the requesting agent uses SC4\* (with SC5\*) to send to the replying agent an encoded address space selection for the ensuing transfer. For a read operation, the address of the selected space is a source of data; for a write operation, the address space is a destination. The requesting agent has four address space options from which to choose: memory, I/O, message, or interconnect. Table 2-4 shows the encoding.

During the reply phase, the replying agent holds SC4\* low to indicate to the requesting agent that it is ready to send or receive data. SC4\* works closely with SC3\* to provide a reply phase handshake for transfer cycles.

**2.3.3.6 SC5\* (Address Space Or Agent Error).** The SC5\* line changes functions depending on the phase of the transfer cycle.

During the request phase, the requesting agent uses SC5\* with SC4\* to select the address space that will be used in the data transfer. For a read operation, the address space is a source of data; for a write operation, the address space is a destination. Table 2-4 lists the encoding for each of the four address spaces.

During the reply phase, the line is encoded with SC6\* and SC7\* to provide four agent error indications. Table 2-5 lists the error codes.

**2.3.3.7 SC6\* (Read-Write Or Agent Error).** The SC6\* line changes functions depending on the phase of the transfer cycle.

During the request phase, the requesting agent uses SC6\* to indicate the direction of data flow for the transfer. When the requesting agent holds SC6\* high, the transfer cycle is a read operation. When the requesting agent holds SC6\* low, the transfer cycle is a write operation.

During the reply phase, SC6\* is encoded with SC5\* and SC7\* to identify agent errors. Refer to the description of SC5\*.

## PARALLEL SYSTEM BUS SPECIFICATION

**2.3.3.8 SC7\* (Reserved Or Agent Error).** The function of SC7\* changes depending on the phase of the transfer cycle.

During the request phase, SC7\* is not used and must be driven high (inactive).

During the reply phase, SC7\* is encoded with SC5\* and SC6\* to identify agent errors. Refer to the description of SC5\*.

**2.3.3.9 SC8\* (Parity On SC7\*-SC4\*).** The function of SC8\* remains the same in both the request phase and the reply phase of the transfer cycle. Agents use SC8\* to provide even parity for the SC7\* through SC4\* signals. As an example, if SC7\* through SC4\* contain an odd number of low signals on the bus, then the agent driving SC7\* through SC4\* must also drive the SC8\* signal low on the bus.

**2.3.3.10 SC9\* (Parity On SC3\*-SC0\*).** The function of SC9\* remains the same in both the request phase and the reply phase of the transfer cycle. Agents use SC9\* to provide even parity for the SC3\* through SC0\* signals. As an example, if SC3\* through SC0\* contain an odd number of low signals on the bus, then the agent driving SC3\* through SC0\* must also drive the SC9\* signal low on the bus.

Table 2-4. Summary Of Functions Of SC9\* Through SC0\*

Signal	Function During Request Phase	Function During Reply Phase
SC0*	Request Phase	Request Phase
SC1*	LOCK	LOCK
SC2*	Data Width	End-of-Cycle
SC3*	Data Width	Requesting Agent Ready
SC4*	Address Space	Replying Agent Ready
SC5*	Address Space	Agent Error
SC6*	Read/Write Operation	Agent Error
SC7*	Reserved	Agent Error
SC8*	Even Parity on SC7*-SC4*	Even Parity on SC7*-SC4*
SC9*	Even Parity on SC3*-SC0*	Even Parity on SC3*-SC0*

# PARALLEL SYSTEM BUS SPECIFICATION

Table 2-5. Functions Of SC9\*--SC0\* During REQUEST Phase

Signal	Driving Agent	Functions During Request Phase															
SC0*	Requesting Agent	Identify transition between request phase and reply phase of the transfer cycle: low = request phase high = not request phase															
SC1*	Requesting Agent	Lock access to the multi-ported resources on the bus: low = lock active high = lock disabled															
SC2* and SC3*	Requesting Agent	Identify the width of the data during the transfer cycle: <table> <tr> <td>SC3*</td> <td>SC2*</td> <td></td> </tr> <tr> <td>high</td> <td>high</td> <td>= 8-bit transfers</td> </tr> <tr> <td>high</td> <td>low</td> <td>= 16-bit transfers</td> </tr> <tr> <td>low</td> <td>high</td> <td>= 24-bit transfers</td> </tr> <tr> <td>low</td> <td>low</td> <td>= 32-bit transfers</td> </tr> </table>	SC3*	SC2*		high	high	= 8-bit transfers	high	low	= 16-bit transfers	low	high	= 24-bit transfers	low	low	= 32-bit transfers
SC3*	SC2*																
high	high	= 8-bit transfers															
high	low	= 16-bit transfers															
low	high	= 24-bit transfers															
low	low	= 32-bit transfers															
SC4* and SC5*	Requesting Agent	Identify the address space for the transfer cycle: <table> <tr> <td>SC5*</td> <td>SC4*</td> <td></td> </tr> <tr> <td>high</td> <td>high</td> <td>= memory address space</td> </tr> <tr> <td>high</td> <td>low</td> <td>= I/O address space</td> </tr> <tr> <td>low</td> <td>high</td> <td>= message address space</td> </tr> <tr> <td>low</td> <td>low</td> <td>= interconnect space</td> </tr> </table>	SC5*	SC4*		high	high	= memory address space	high	low	= I/O address space	low	high	= message address space	low	low	= interconnect space
SC5*	SC4*																
high	high	= memory address space															
high	low	= I/O address space															
low	high	= message address space															
low	low	= interconnect space															
SC6*	Requesting Agent	Identify the type of operation: low = write operation high = read operation															
SC7*	Requesting Agent	Not used; must be high.															
SC8*	Requesting Agent	Provide even parity for SC7* through SC4*: low = odd number of low bits on SC7* through SC4*. high = even number of low bits on SC7* through SC4*															
SC9*	Requesting Agent	Provide even parity for SC3* through SC0*: low = odd number of low bits on SC3* through SC0*. high = even number of low bits on SC3* through SC0*.															



# PARALLEL SYSTEM BUS SPECIFICATION

Table 2-6. Functions Of SC9\*-SC0\* During REPLY Phase

Signal	Driving Agent	Functions During Reply Phase Of a Transfer Cycle																											
SC0*	Requesting Agent	Identify transition between request phase and reply phase of the transfer cycle: low = request phase high = not request phase																											
SC1*	Requesting Agent	Lock access to the dual port resources on the bus: low = lock active high = lock disabled																											
SC2*	Requesting Agent	Place an end-of-cycle indication onto the bus: low = end-of-cycle indication for current transfer cycle. high = not end-of-cycle.																											
SC3*	Requesting Agent	Provide a requesting-agent-ready indication on the bus (part of the reply phase handshake): low = agent ready to conduct a data transfer operation. high = agent not ready.																											
SC4*	Replying Agent	Provide a replying-agent-ready indication on the bus (part of the reply phase handshake): low = agent ready to conduct a data transfer operation. high = agent not ready.																											
SC5* SC6* and SC7*	<table> <tr> <th>SC5*</th><th>SC6*</th><th>SC7*</th></tr> <tr> <td>low</td><td>low</td><td>low</td></tr> <tr> <td>low</td><td>low</td><td>high</td></tr> <tr> <td>low</td><td>high</td><td>low</td></tr> <tr> <td>low</td><td>high</td><td>high</td></tr> <tr> <td>high</td><td>low</td><td>low</td></tr> <tr> <td>high</td><td>low</td><td>high</td></tr> <tr> <td>high</td><td>high</td><td>low</td></tr> <tr> <td>high</td><td>high</td><td>high</td></tr> </table>	SC5*	SC6*	SC7*	low	low	low	low	low	high	low	high	low	low	high	high	high	low	low	high	low	high	high	high	low	high	high	high	Place an agent error indication onto the bus:  Reserved. Transfer-not-understood error; multiple errors sensed in transfer. Agent data error. Width error; width of transfer is not compatible with replying agent. Reserved. Continuation error; replying agent is unable to continue an operation. NACK; Replying agent cannot respond to message operation. No error; transfer completed without errors.
SC5*	SC6*	SC7*																											
low	low	low																											
low	low	high																											
low	high	low																											
low	high	high																											
high	low	low																											
high	low	high																											
high	high	low																											
high	high	high																											

# PARALLEL SYSTEM BUS SPECIFICATION

Table 2-6. Functions Of SC9\*-SC0\* During REPLY Phase (continued)

Signal	Driving Agent	Functions During Reply Phase Of a Transfer Cycle
SC8*	Reply Agent	Provide even parity for SC7* through SC4*: low = odd number of low bits on SC7* through SC4*. high = even number of low bits on SC7* through SC4*.
SC9*	Requesting Agent	Provide even parity for SC3* through SC0*: low = odd number of low bits on SC3* through SC0*. high = even number of low bits on SC3* through SC0*.

## 2.3.4 Exception Cycle Signal Group

The Parallel System bus provides a group of two signals, the exception cycle signal group for passing indications of exception errors to all agents. The two exception error signals are bus error (BUSERR\*) and bus timeout (TIMOUT\*). Each of these signals is on a dedicated line on the Parallel System bus.

**2.3.4.1 BUSERR\* EXCEPTION.** An agent activates the bus error signal to indicate detection of a data integrity problem during a transfer. Problems reported through the BUSERR\* signal are detection of a parity error on the AD31\* through ADO\* lines or on the SC9\* through SC0\* lines. BUSERR\* is received by all agents and generated by all agents that can detect transfer integrity problems.

**2.3.4.2 TIMOUT\* EXCEPTION.** An agent uses the timeout signal to indicate when a bus timeout condition occurs. Bus timeout on the Parallel System bus is a result of the CSM determining that an agent is taking too much time to respond to a handshake signal on the bus.

Either type of agent can cause the CSM to activate TIMOUT\*. If a requesting agent does not present the next data transfer quickly enough after a handshake from a replying agent, then the CSM activates TIMOUT\*. If a replying agent does not respond quickly enough with its portion of the reply phase handshake in a transfer cycle, then the CSM activates TIMOUT\*.

TIMOUT\* is received by all active agents on the bus and is generated from timeout circuitry located on the CSM.

## PARALLEL SYSTEM BUS SPECIFICATION

### 2.3.5 Central Control Signal Group

The central control signal group provides status concerning the operating state of the entire system. The signal group consists of 7 logic signals for the Parallel System bus. The system control signal group consists of RST\*, RSTNC\*, DCLOW\*, PROT\*, BCLK\*, CCLK\*, and LACHn\*. Each of the signals is described in more detail in the following paragraphs.

**2.3.5.1 RST\* (Reset).** The CSM drives RST\* as a system-level initialization signal to all agents. All agents receive the RST\* signal and the CSM drives the RST\* signal. Agents may monitor RST\* with DCLOW\* and PROT\* to detect power failures.

**2.3.5.2 RSTNC\* (Reset-Not-Completed).** Agents generate RSTNC\* onto the bus to extend the initialization time-period provided by the CSM. The RSTNC\* signal holds other agents from starting bus operations. RSTNC\* is an OR-tied signal that is low when one or more agents on the Parallel System bus have not completed their reset requirements. Agents cannot perform bus cycles while RSTNC\* is active. RSTNC\* must be driven on all agents that need more initialization time than the RST\* signal provides. RSTNC\* must be received by all agents.

**2.3.5.3 DCLOW\* (DC Power LOW).** The CSM provides an active DCLOW\* signal to all agents as a pre-warning of an imminent power failure. The CSM monitors the power level from the power supply and holds DCLOW\* low during normal system operation. When the power supply identifies a power fail for the CSM, the CSM activates DCLOW\* to indicate to all agents that a power failure is imminent. Systems use DCLOW\* as the control signal for providing back-up power. The DCLOW\* signal is asynchronous to the clock.

**2.3.5.4 PROT\* (Protect).** The CSM provides an active PROT\* signal as a power failure protection signal to other system agents. PROT\* is a delayed result of the CSM sensing a power failure via DCLOW\*. All agents in the system receive PROT\* if they provide battery backed-up resources.

**2.3.5.5 BCLK\* (Bus Clock).** Only the CSM generates BCLK\* and all agents in a system receive BCLK\*. An agent uses the bus clock signal to drive the arbitration and timing state machines on the Parallel System bus. BCLK\* has a maximum frequency of ten megahertz on the bus. The falling edge of BCLK\* provides all system timing references.

## PARALLEL SYSTEM BUS SPECIFICATION

**2.3.5.6 CCLK\* (Central Clock).** The central clock signal is a fixed frequency clock which may be used for additional timing among agents on the Parallel System bus. CCLK\* operates with a specified relationship to BCLK\* and twice the frequency.

**2.3.5.7 LACHn\* (ID Latch).** An agent uses its ID latch signal only during the power-up initialization or after a reset. The LACHn\* signal is driven by the CSM and serves two functions: it tells the agent when to latch the arbitration ID and when to latch the cardslot ID from the ARB4\* through ARB0\* lines.

The ID latch signal is called LACHn\* (where "n" is the cardslot to which the ID is assigned). Each of the 20 possible cardslots contains a LACHn\* signal that is activated when the CSM drives one of the address/data (AD19\*-AD0\*) lines. As an example, the agent in cardslot 7 accepts its ID when the CSM drives AD7\* which is connected to LACH7\*.

### 2.3.6 Power

The Parallel System bus provides power for bus agents. The system power includes dc power at +5 volts, +12 volts, -12 volts, ground, and facilities for +5 volts battery back-up. More details are presented in section 4 of this specification.



## PARALLEL SYSTEM BUS SPECIFICATION

### 3. PARALLEL SYSTEM BUS PROTOCOL

#### 3.1 GENERAL

This section uses the signals presented in section 2 to describe the protocol on the Parallel System bus. The description of the protocol is organized according to the three cycles that occur on the bus: the arbitration cycle, the transfer cycle, and the exception cycle. The protocol for each cycle is presented in two stages, first in timing diagrams and then in state-flow diagrams.

The timing diagrams provide a time-ordered representation of bus operations by showing the transition relationships among the signals that create the bus protocol. The timing diagrams illustrate the synchronous nature of the Parallel System bus by showing the bus clocks as vertical lines. The bus clock provides to all agents a common signal for synchronizing operations.

State-flow diagrams present the lowest level of detail in defining the protocol responsibilities of each agent in an operation. The state-flow diagrams list signal conditions required for each state-transition in a cycle.

Figure 3-1 shows the three types of bus cycles and the signal groups that create the protocol for each cycle. Requesting agents initiate arbitration cycles, transfer cycles, and exception cycles. Replying agents can only respond to transfer cycles and initiate exception cycles when they sense errors on the bus.

The arbitration cycle consists of a resolution phase to select a bus owner and an acquisition phase to give bus control to the selected owner. The transfer cycle consists of a request phase for passing command/address information and a reply phase for passing handshake/data information. The exception cycle consists of a signal phase that identifies an error condition on the bus and terminates all other types of bus cycles, and a recovery phase that provides a predetermined amount of idle time.

The communication protocol among agents on the Parallel System bus is consistent for all operations. Requesting and replying agents implement the protocol by controlling groups of signal lines. To simplify the discussion, the protocol description is presented in three steps; the arbitration cycle is first, followed by the transfer cycle, and finally, the exception cycle.

# PARALLEL SYSTEM BUS SPECIFICATION

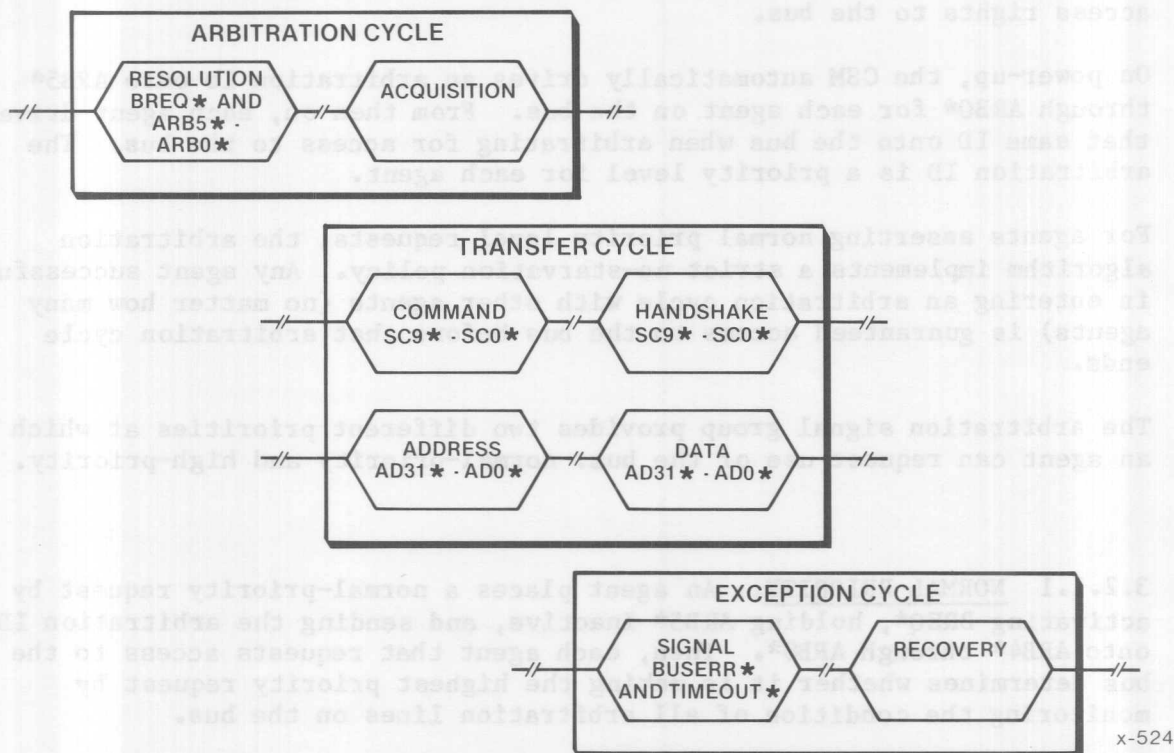


Figure 3-1. Bus Cycle Details

## 3.2 ARBITRATION CYCLE OVERVIEW

An agent that wishes to transfer data on the Parallel System bus must begin by performing an arbitration cycle. During an arbitration cycle, one or more agents arbitrate for control of the bus. As a result of the arbitration cycle, one agent gains control of the bus and becomes the bus owner. Agents must gain control of the Parallel System bus before they can transfer data.

The arbitration cycle performs two functions: first, it allows all agents equal opportunity to access the bus, and second, it eliminates the possibility of more than one agent trying to transfer data on the bus at any one instant. In a case where more than one agent requests access to the bus at the same instant, the arbitration cycle grants sequential access to the agents based on their priority.

## PARALLEL SYSTEM BUS SPECIFICATION

### 3.2.1 Bus Arbitration Priority

All agents in a system share access to six signal lines on the bus, ARB5\* through ARB0\*. These signals provide a feature called the arbitration ID that allows agents to decide among themselves which agent has first access rights to the bus.

On power-up, the CSM automatically drives an arbitration ID onto ARB5\* through ARB0\* for each agent on the bus. From then on, each agent drives that same ID onto the bus when arbitrating for access to the bus. The arbitration ID is a priority level for each agent.

For agents asserting normal priority level requests, the arbitration algorithm implements a strict no-starvation policy. Any agent successful in entering an arbitration cycle with other agents (no matter how many agents) is guaranteed access to the bus before that arbitration cycle ends.

The arbitration signal group provides two different priorities at which an agent can request use of the bus: normal-priority and high-priority.

**3.2.1.1 NORMAL PRIORITY.** An agent places a normal-priority request by activating BREQ\*, holding ARB5\* inactive, and sending the arbitration ID onto ARB4\* through ARB0\*. Then, each agent that requests access to the bus determines whether it is making the highest priority request by monitoring the condition of all arbitration lines on the bus.

If the ID on ARB4\* through ARB0\* matches the arbitration ID of the agent, the agent becomes a bus owner. If the condition of the arbitration lines does not match that of the agent, the agent stops driving the arbitration ID lines below the bit that doesn't match, and waits for the next resolution phase in the arbitration cycle.

**3.2.1.2 HIGH PRIORITY.** In applications where the CSM-assigned "priority" (the arbitration ID) does not allow enough discrimination among agents, an agent uses the high-priority feature.

The protocol requires that agents making a high priority request (ARB5\* active) obtain access to the bus before those using normal priority. When a high-priority request enters during an arbitration cycle, the request immediately enters the next resolution phase of the arbitration cycle rather than waiting for the next arbitration cycle as do normal-priority requests.

An agent places a high-priority request by holding ARB5\* active while sending the arbitration ID onto the bus. By activating the ARB5\* signal, an agent guarantees itself access to the bus before any simultaneous requests from other agents asserting a normal-priority request. When more than one agent simultaneously places a high-priority request, the agent with an active ARB5\* and with the higher priority within ARB4\* through ARB0\* gains first access.

## PARALLEL SYSTEM BUS SPECIFICATION

Table 3-1 summarizes the interaction between agents of different priority levels on bus.

Table 3-1. Arbitration Priority Protocol

Priority Level Of Requesting Agent A	Priority Level Of Requesting Agent B	Next Bus Owner	Description
Normal	Normal	either	Arbitration based on ARB4* through ARB0* to select next bus owner.
Normal	High	Agent B	Allow the high priority agent to be the next bus owner.
High	Normal	Agent A	Allow the high priority agent to be the next bus owner.
High	High	either	Arbitrate among high priority agents based on ARB4* through ARB0*.

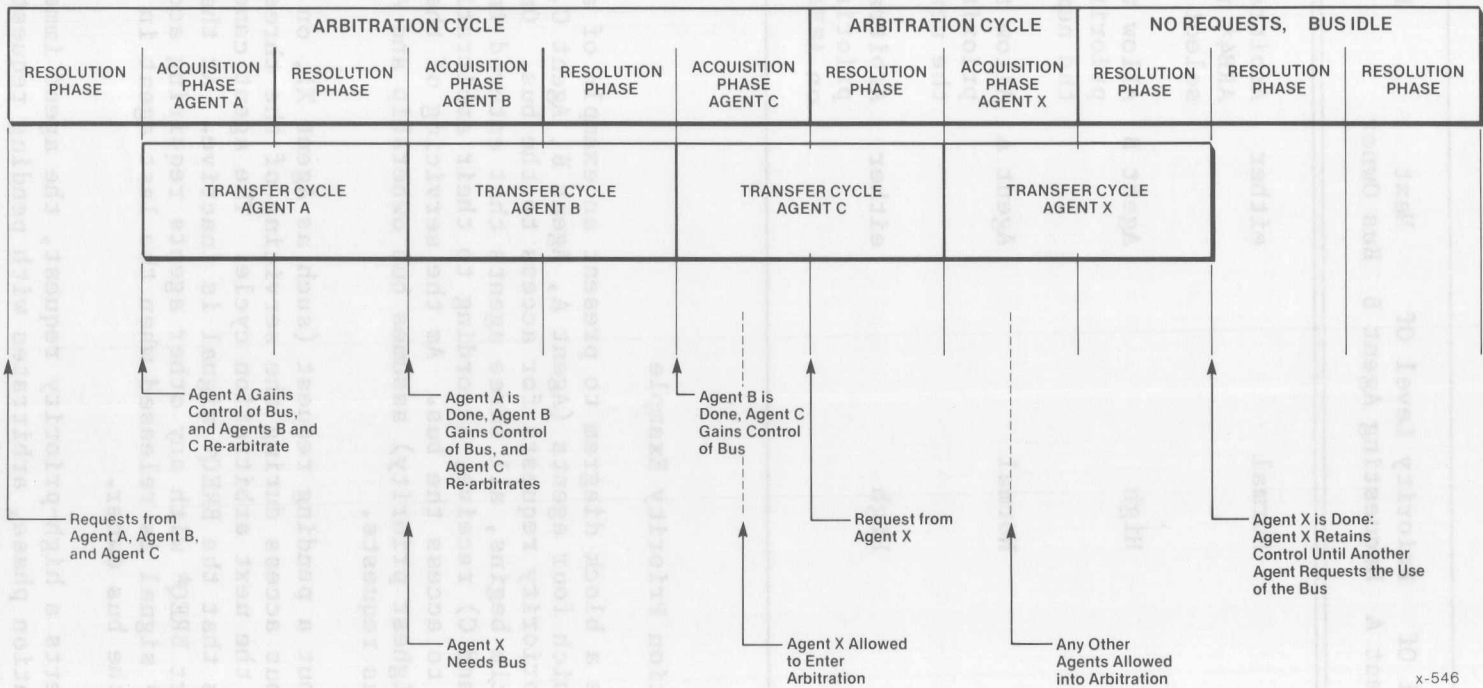
### 3.2.2 Arbitration Priority Example

Figure 3-2 uses a block diagram to present an example of an arbitration cycle during which four agents (Agent A, Agent B, Agent C, and Agent X) assert normal-priority requests for access to the bus. Once the first arbitration cycle begins, all three agents that entered into arbitration (Agents A, B, and C) receive, according to their arbitration ID priority, an opportunity to access the bus. As the servicing of the three begins, Agent A (the highest priority) assumes bus ownership and Agents B and C have pending bus requests.

Any agent without a pending request (such as Agent X), on determining that it needs bus access during the servicing of the three agents, is held-off until the next arbitration cycle. The agent cannot assert BREQ\* until it senses that the BREQ\* signal is inactive. At that time, the agent may assert BREQ\* with any other agents requiring access to the bus. The BREQ\* signal is released when the last agent in the arbitration cycle becomes the bus owner.

If Agent X asserts a high-priority request, the agent immediately enters the next resolution phase, arbitrates with pending requests, and gains next-access rights in place of Agent C; access for Agent C is delayed until Agent X is done. As an example, if Agent X were to assert a high-priority request at the point on Figure 3-2 labeled "Agent X needs bus", then Agent X would gain control of the bus at the point where the access for Agent C is shown.





x-546

Figure 3-2. Typical Arbitration Cycle Sequence

## PARALLEL SYSTEM BUS SPECIFICATION

### 3.2.3 Arbitration ID And Cardslot ID Assignment

On power-up, the CSM in each Multibus II system assigns cardslot ID's (0 through 19) to each agent and assigns arbitration ID's to each agent on the bus. The sequence involves the use of the arbitration ID lines (ARB5\* through ARB0\*), the address/data bus lines (AD19\* through AD0\*), the ID latch signal (LACHn\*), and the reset signal (RST\*). When ARB5\* is low, the CSM has cardslot IDs on ARB4\* through ARB0\*; when ARB5\* is high, the CSM has arbitration IDs on ARB4\* through ARB0\*.

The default assignment of arbitration IDs and cardslot IDs is listed in Table 3-2.

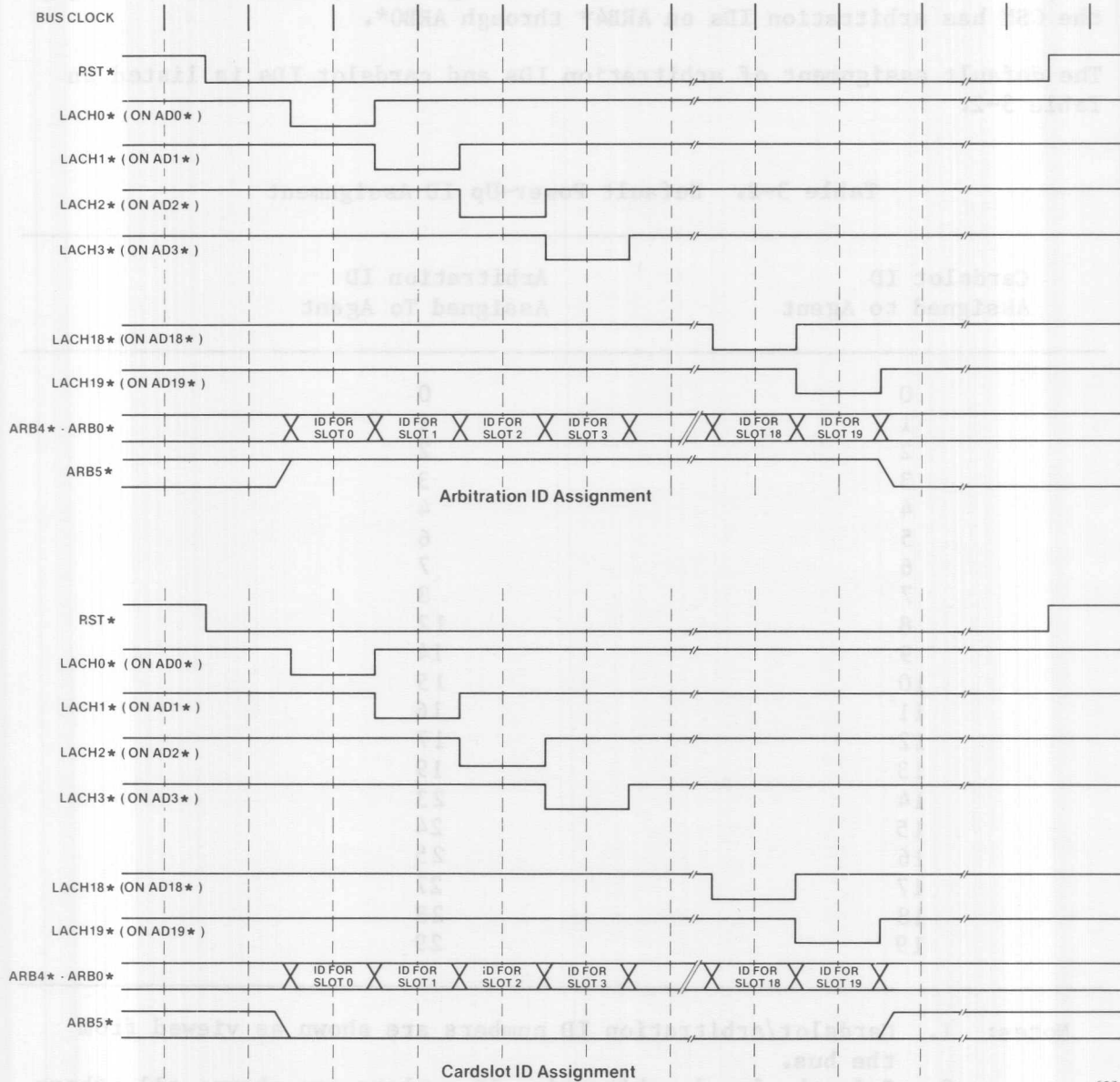
Table 3-2. Default Power-Up ID Assignment

Cardslot ID Assigned to Agent	Arbitration ID Assigned To Agent
0	0
1	1
2	2
3	3
4	4
5	6
6	7
7	8
8	12
9	14
10	15
11	16
12	17
13	19
14	23
15	24
16	25
17	27
18	28
19	29

- Notes:
1. Cardslot/arbitration ID numbers are shown as viewed from the bus.
  2. Only the legal arbitration ID numbers are shown; all others are illegal.
  3. Another arbitration ID may be assigned to an agent, but the Cardslot ID also changes as shown in the table.

Figure 3-3 shows the timing sequence for a CSM assigning IDs to each agent and cardslot in the system. When RST\* is low and ARB5\* is low, the CSM is assigning cardslot IDs to all agents on the bus. While RST\* is still low, the CSM drives ARB5\* high to assign the arbitration IDs to all agents on the bus.

# PARALLEL SYSTEM BUS SPECIFICATION



x-557

Figure 3-3. Arbitration/Cardslot ID Assignment Timing Sequence

## PARALLEL SYSTEM BUS SPECIFICATION

With each arbitration/cardslot ID, the CSM drives a corresponding address/data line (connected to the LACHn\* signal for each cardslot) to identify the destination for the ID.

The arbitration ID for the agent in cardslot 0 recognizes ARB4\* through ARB0\* when they are all low (00000). The agent in cardslot 19 recognizes ARB4\* through ARB0\* as being 11101.

Each agent uses the address/data line to identify the ID from the ARB4\* through ARB0\* lines and latch it within a register.

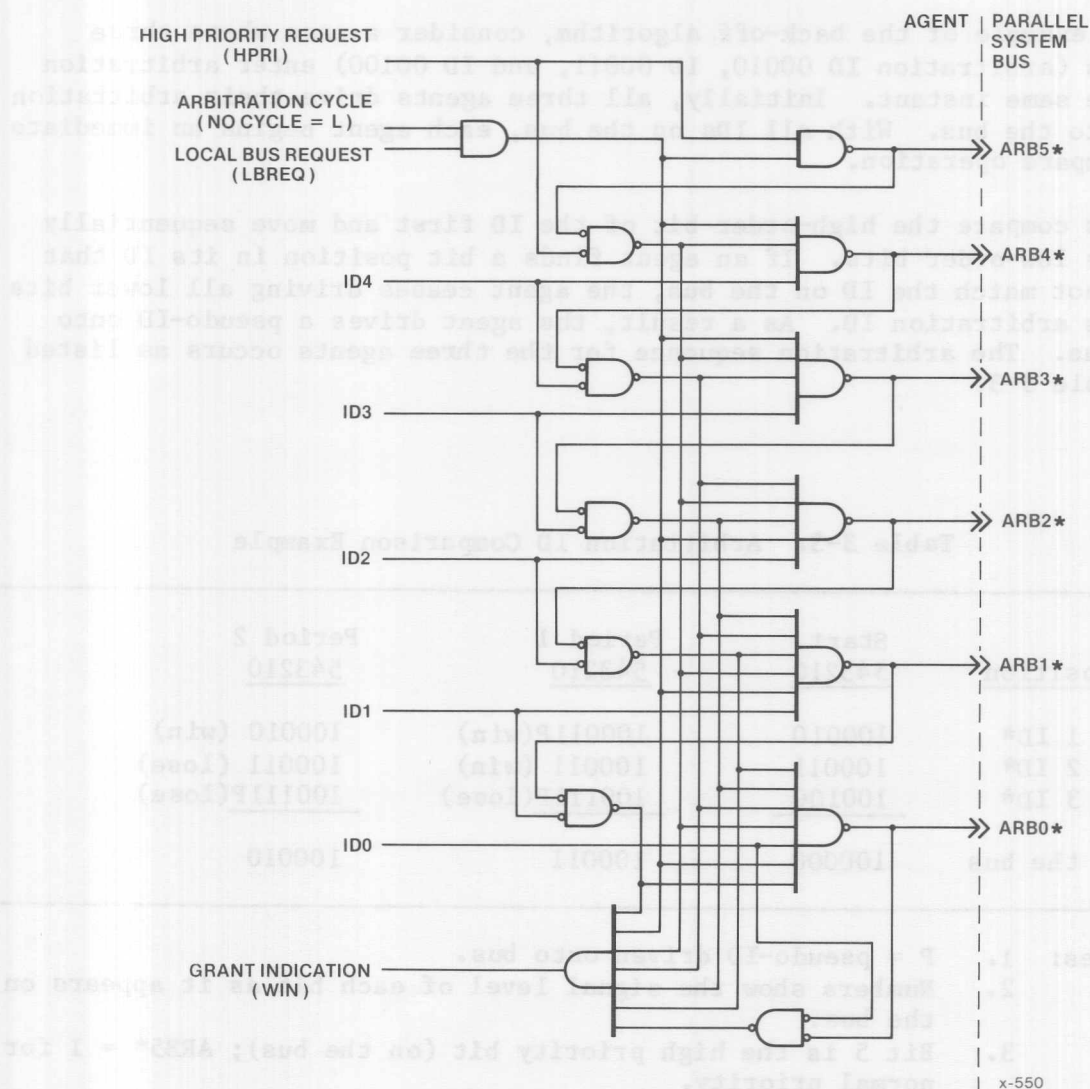


Figure 3-4. Arbitration ID Interface Example (At Each Agent)



## PARALLEL SYSTEM BUS SPECIFICATION

### 3.2.4 Arbitration Sequence

An agent uses its arbitration ID to arbitrate for bus access of three operations. The function of the arbitration ID is described further in the following paragraphs.

When an agent attempts to gain control of the bus, the agent arbitrates by driving its arbitration ID onto the bus and comparing bit-by-bit with the other agent IDs on the bus. At the point where the agent ID does not match the ID on the bus, the agent uses a back-off algorithm.

If an agent drives a low signal level onto an arbitration line, that signal level is dominant in forming the bus ID; the bus ID consists of a logical "AND" function performed on all agent IDs.

As an example of the back-off algorithm, consider a case where three agents (arbitration ID 00010, ID 00011, and ID 00100) enter arbitration at the same instant. Initially, all three agents drive their arbitration ID onto the bus. With all IDs on the bus, each agent begins an immediate ID compare operation.

Agents compare the high-order bit of the ID first and move sequentially to the low-order bits. If an agent finds a bit position in its ID that does not match the ID on the bus, the agent ceases driving all lower bits of its arbitration ID. As a result, the agent drives a pseudo-ID onto the bus. The arbitration sequence for the three agents occurs as listed in Table 3-3.

Table 3-3. Arbitration ID Comparison Example

Bit Position	Start 543210	Period 1 543210	Period 2 543210
Agent 1 ID*	100010	100011P(win)	100010 (win)
Agent 2 ID*	100011	100011 (win)	100011 (lose)
Agent 3 ID*	100100	100111P(lose)	100111P(lose)
ID on the bus	100000	100011	100010

- Notes:
1. P = pseudo-ID driven onto bus.
  2. Numbers show the signal level of each bit as it appears on the bus.
  3. Bit 5 is the high priority bit (on the bus); ARB5\* = 1 for normal priority.

## PARALLEL SYSTEM BUS SPECIFICATION

The ID comparison allows each agent to make an absolute determination as to whether or not it is the highest priority requesting agent. The ID-comparison periods occur asynchronous to the bus clock.

Agents sequentially re-check, in a most-significant to least-significant sequence, each bit of the agent ID that does not match the bus ID. When an agent senses a non-compare at a particular bit position, that agent loses the arbitration and must re-check after sensing EOC on the bus. Agents that lose during the arbitration sequence continue to drive an arbitration ID onto the bus. However, the ID is not the true ID of the agent, but rather the pseudo ID created by not driving the bits that do not compare.

Losing agents continue to hold active the BREQ\* signal to hold other agents out of the arbitration cycle. When the losing agents sense an active EOC handshake, they enter the next acquisition phase of an arbitration cycle and repeat the arbitration process, but this time with one less agent.

Figure 3-4 shows an example of arbitration circuit on a requesting agent, and Figure 3-5 shows the timing sequence for a requesting agent asserting a bus request (BREQ\*) and placing its arbitration ID onto the bus. As the figure implies, the resolution phase of the arbitration cycle is always three clock cycles.

An agent may enter itself into the resolution phase of an arbitration cycle when all agents that drove BREQ\* active (from the previous resolution phase) are serviced. As each is serviced, it stops driving BREQ\* (an OR-tied signal on the bus). When all agents in the resolution phase are serviced, BREQ\* goes inactive on the bus. One clock after BREQ\* is inactive, agents may begin asserting BREQ\* for the next arbitration cycle. At that point, agents assert ARB5\* through ARB0\* to enter the next resolution phase.

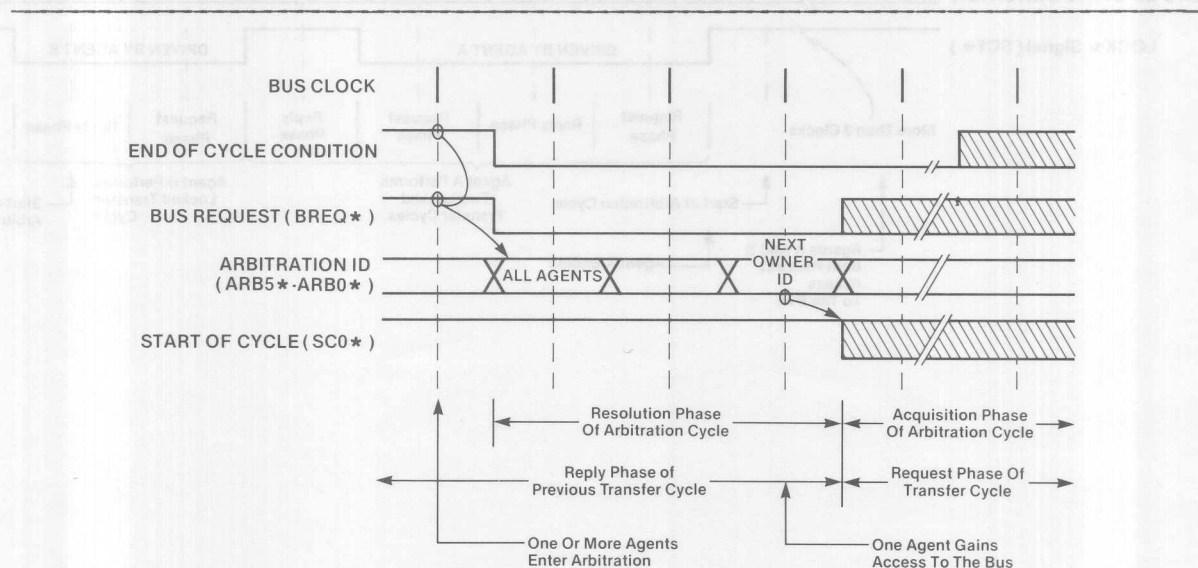


Figure 3-5. Timing Sequence For Bus Control Acquisition

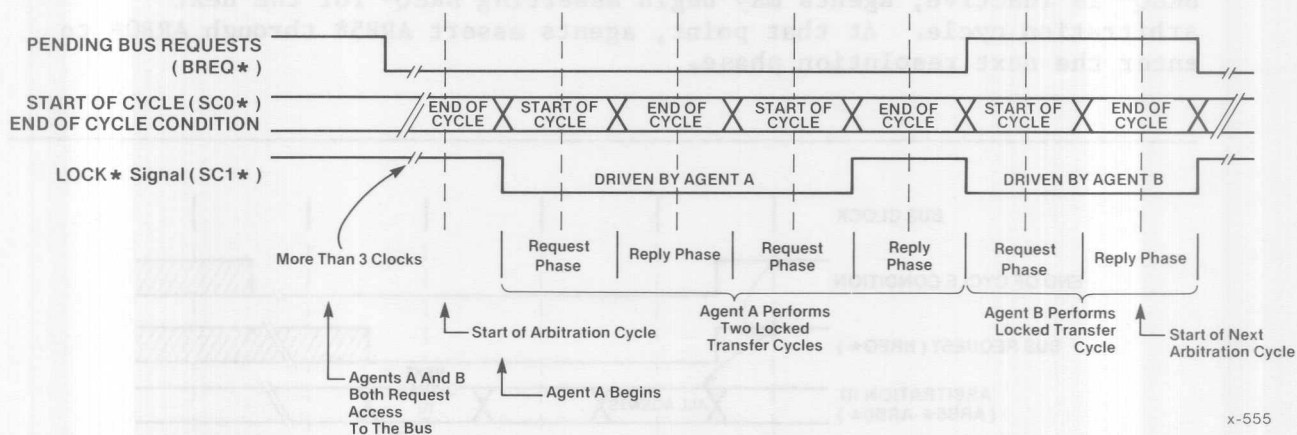
## PARALLEL SYSTEM BUS SPECIFICATION

**3.2.4.2 BUS OWNERSHIP.** When an agent recognizes that its arbitration ID matches the ID on the bus, the agent begins an acquisition phase.

During the acquisition phase, the bus owner performs transfer cycles and all agents with pending requests begin arbitrating again to determine the next bus owner. During the last reply phase of the transfer cycle, the bus owner sends an EOC\* signal onto the bus (via SC2\*). Then, as the current bus owner releases control of the bus (if another agent request the bus, that agent immediately enters a bus acquisition phase.

A bus owner can perform several consecutive transfer cycles by asserting the LOCK\* signal on SC1\*. In doing so, the bus owner retains ownership of the bus for more than one transfer cycle. By asserting LOCK\*, the bus owner guarantees itself exclusive access to the bus resources until it releases LOCK\*; other agents are prohibited from gaining ownership of the bus.

Figure 3-6 shows the timing sequence for an operation that locks the bus. During the time that LOCK\* is asserted, the bus owner performs several consecutive transfer cycles without interruption from any other devices. Agents cannot transfer ownership of the bus when the bus is locked.



x-555

Figure 3-6. Timing Sequence With Bus Locked

## PARALLEL SYSTEM BUS SPECIFICATION

**3.2.4.3 BUS RELEASE.** A bus owner releases control of the bus only on generating an EOC\* signal when LOCK\* is inactive. The EOC\* handshake indicates to all agents that the bus is available and allows the next bus owner to acquire control of the bus.

If the current bus owner is the last agent in the arbitration cycle to gain access to the bus, then as the current bus owner stops driving the BREQ\* signal, the signal goes inactive on the bus. On going inactive, BREQ\* allows all agents to begin the next arbitration cycle. As the bus owner completes its operation, its EOC\* indication on the bus allows the next bus owner to assume ownership of the bus.

If no agent requests access to the bus, the last agent to be the bus owner retains ownership of the bus. As such, an agent can perform another transfer cycle without arbitrating for access to the bus.

Agents extend the duration of a resolution phase while a bus owner performs a transfer cycle that transfers multiple data. Figure 3-7 shows the timing for the sequence.

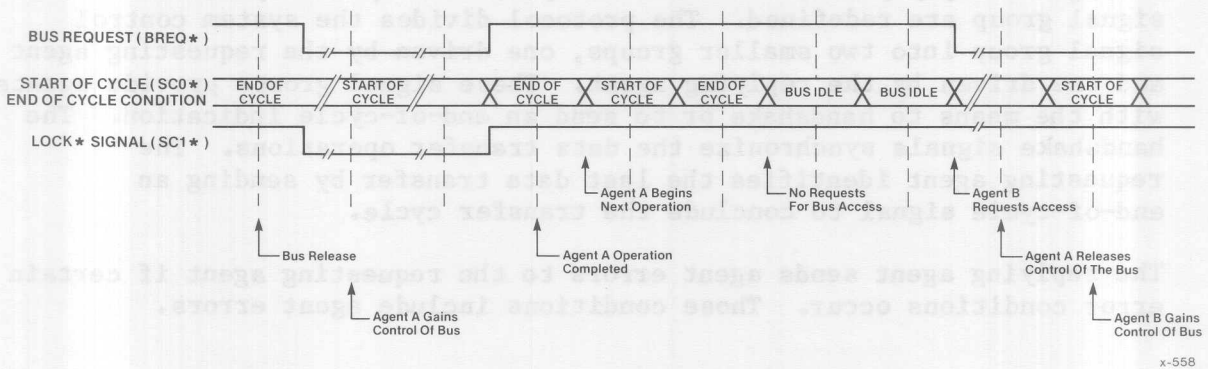


Figure 3-7. Timing Sequence For Bus Release



## PARALLEL SYSTEM BUS SPECIFICATION

### 3.3 TRANSFER CYCLE OVERVIEW

An agent must complete a successful arbitration cycle before it can perform a transfer cycle. On completing an arbitration cycle, one agent gains ownership of the bus and starts a transfer cycle immediately. Only the agent that owns the bus may conduct transfer cycles on the Parallel System bus.

An agent performs a transfer cycle by manipulating signals within two signal groups on the Parallel System bus: the Address/Data Signal Group and the System Control Signal Group.

The address/data signal group (AD31\* through AD0\*) is a set of 32 signal lines that contains address information during the request phase and data information during the reply phase of a transfer cycle. In other words, the function of AD31\* through AD0\* changes depending on the phase of the transfer cycle.

The system control signal group (SC9\* through SC0\*) is a set of 9 signals that define the operation to be performed in the transfer cycle. As with the address/data lines, the function performed by each of the SC9\* through SC0\* lines is phase-dependent. The signals may mean one thing during a request phase of a transfer cycle and something different during a reply phase of an operation.

In the request phase, the requesting agent uses the system control signal group to notify the replying agent of the address space, the data width, and the command for the ensuing data transfer operation. The requesting agent also uses the address/data signal group to transfer the address for the operation.

During the reply phase, the functions performed by the system control signal group are redefined. The protocol divides the system control signal group into two smaller groups, one driven by the requesting agent and one driven by the replying agent. These signal groups provide agents with the means to handshake or to send an end-of-cycle indication. The handshake signals synchronize the data transfer operations. The requesting agent identifies the last data transfer by sending an end-of-cycle signal to conclude the transfer cycle.

The replying agent sends agent errors to the requesting agent if certain error conditions occur. Those conditions include agent errors.

#### 3.3.1 Types of Transfer Cycles

There are three basic types of transfer cycle: the single-transfer operation, the sequential-transfer operation, and the broadcast operation. Each type implements a slightly different handshake sequence. Each is described further in the following paragraphs.

**3.3.1.1 SINGLE-TRANSFER OPERATION.** The single-transfer operation consists of one transfer cycle. In the cycle, the requesting agent performs one data transfer with one handshake on SC3\* and sends an end-of-cycle indication on SC2\*. To complete the handshake, the replying agent provides SC4\*.

During a normal read operation, the replying agent provides data on the AD31\* through AD0\* lines. The replying agent asserts the REPLIER READY signal on SC4\* when it has placed valid data on the AD31\* through AD0\* lines. The requesting agent asserts EOC on SC2\* and REQUESTOR READY on SC3\* when it is ready to accept data from the bus. When both agents hold SC3\* and SC4\* active, the handshake occurs and the data transfer occurs synchronous with the clock edge.

Figure 3-8 shows a block diagram of the read operation and Figure 3-9 shows the signal sequences for the read operation handshake.

During a normal write operation, the requesting agent provides data on the AD31\* through AD0\* lines. The requesting agent asserts the EOC on SC2\* and the REQUESTOR READY signal on SC3\* when it has placed valid data on the AD31\* through AD0\* lines. The replying agent asserts REPLIER READY on SC4\* when it has accepted data from the bus. When both agents hold SC3\* and SC4\* active, the handshake occurs and the data transfer occurs synchronous with the clock edge.

Figure 3-10 shows a block diagram of the write operation and Figure 3-11 shows the signal sequences for the write operation handshake.

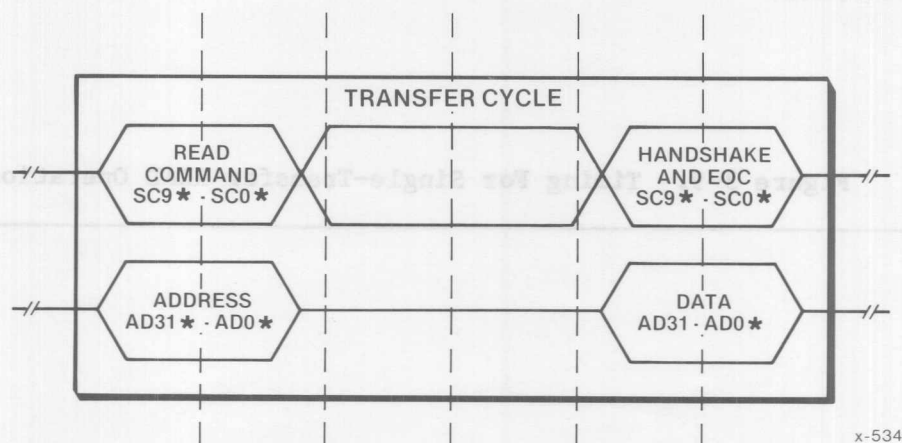


Figure 3-8. Block Diagram of Single-Transfer READ Operation

# PARALLEL SYSTEM BUS SPECIFICATION

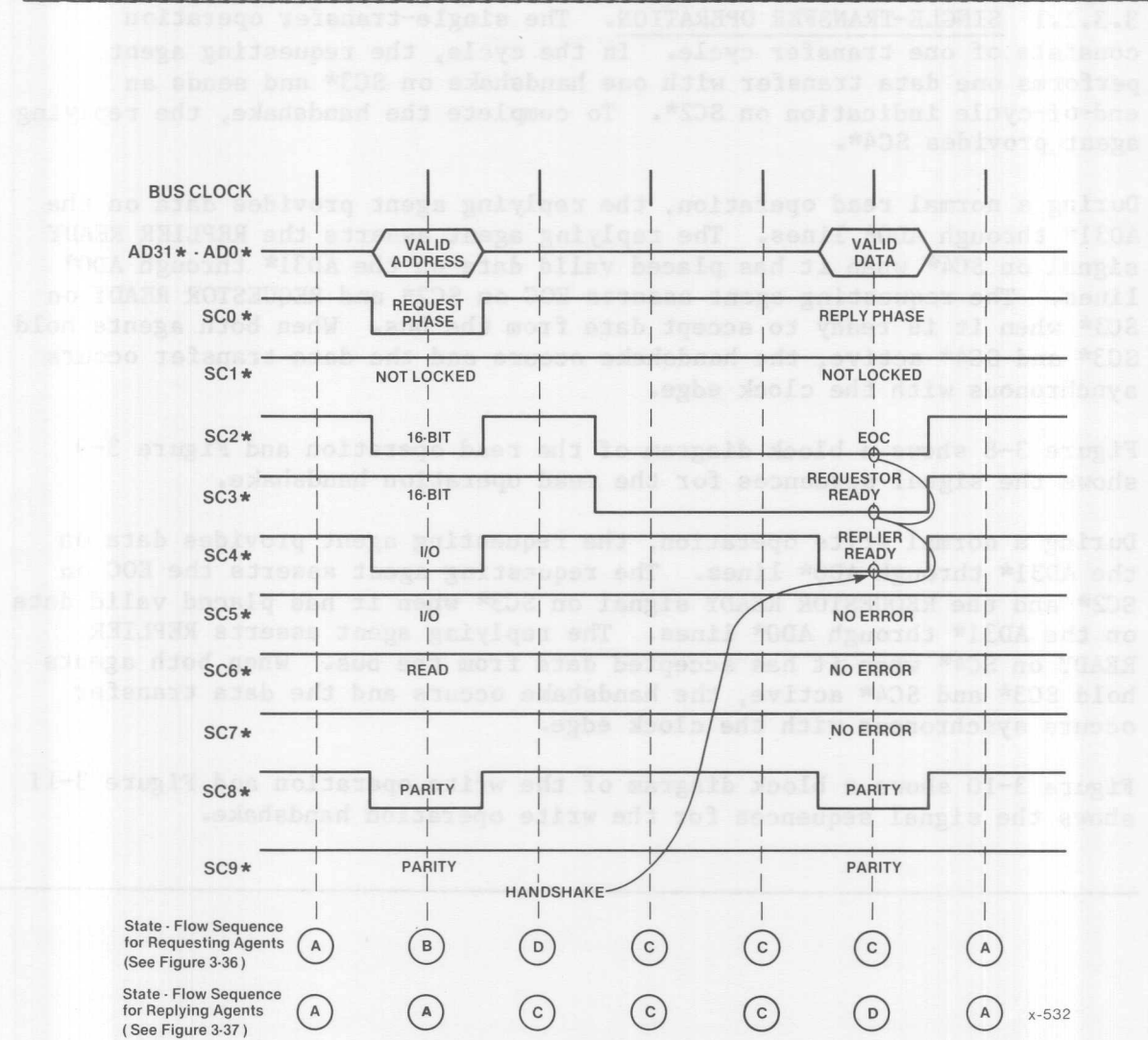


Figure 3-9. Timing For Single-Transfer READ Operation

# PARALLEL SYSTEM BUS SPECIFICATION

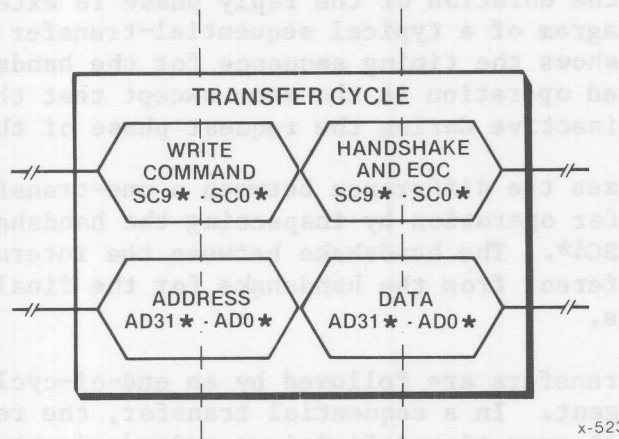


Figure 3-10. Block Diagram of Single-Transfer WRITE Operation

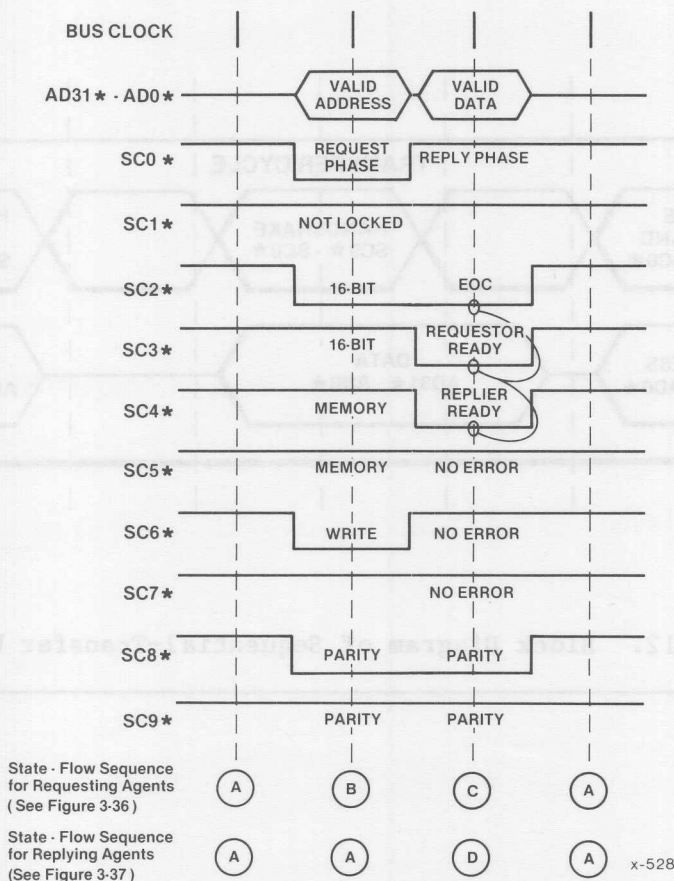


Figure 3-11. Timing For Single-Transfer WRITE Operation



## PARALLEL SYSTEM BUS SPECIFICATION

**3.3.1.2 SEQUENTIAL-TRANSFER OPERATION.** The sequential-transfer differs from the single operation in that the operation consists of one transfer cycle, however, the duration of the reply phase is extended. Figure 3-12 shows a block diagram of a typical sequential-transfer write operation and Figure 3-13 shows the timing sequence for the handshake. The timing diagram for a read operation is the same except that the requesting agent is holding SC6\* inactive during the request phase of the transfer cycle.

An agent recognizes the difference between a one-transfer operation and a sequential transfer operation by inspecting the handshake signals on SC2\*, SC3\*, and SC4\*. The handshake between the intermediate data transfers is different from the handshake for the final data transfer, as Figure 3-13 shows.

The sequential transfers are followed by an end-of-cycle indication from the requesting agent. In a sequential transfer, the replying agent must retain the initial operation definitions gained via the transfer control signal group during the request phase of the transfer cycle. That information applies to all data transfers in the sequential operation.

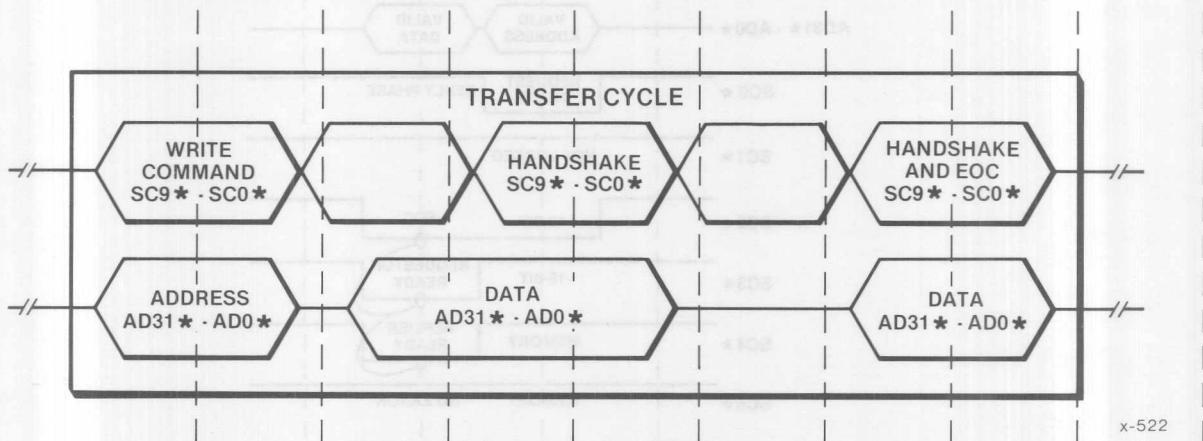


Figure 3-12. Block Diagram of Sequential-Transfer WRITE Operation

# PARALLEL SYSTEM BUS SPECIFICATION

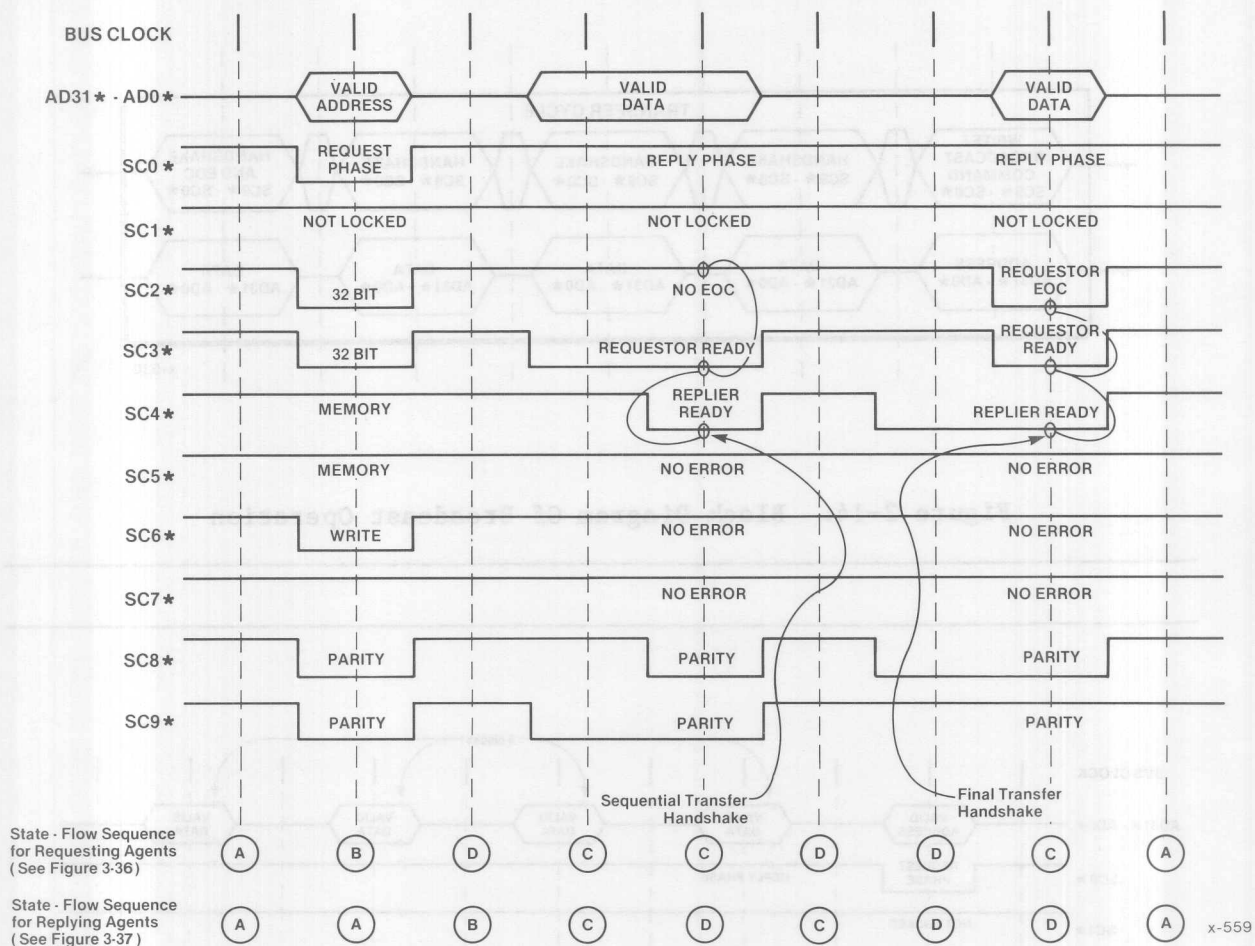


Figure 3-13. Timing For Sequential-Transfer WRITE Operation

**3.3.1.3 MESSAGE BROADCAST OPERATION.** The broadcast operation differs from the single-transfer operation in two areas. First, the request phase addresses multiple replying agents rather than only one agent. Second, the operation can only write data to the replying agents; it cannot be used to read data. Figure 3-14 shows a diagram of a broadcast operation and Figure 3-15 shows the timing sequence for the handshake.

The broadcast operation is only for message space operations and requires a destination address of 00H, as seen from the bus.

In a broadcast, the replying agents do not respond with the normal handshake signal (on SC4\*). Instead, the requesting agent signals that data is valid and waits for 8 clocks. The requesting agent also provides an EOC signal (SC2\*) during the final transfer in the broadcast. The handshake is highlighted in Figure 3-15.

# PARALLEL SYSTEM BUS SPECIFICATION

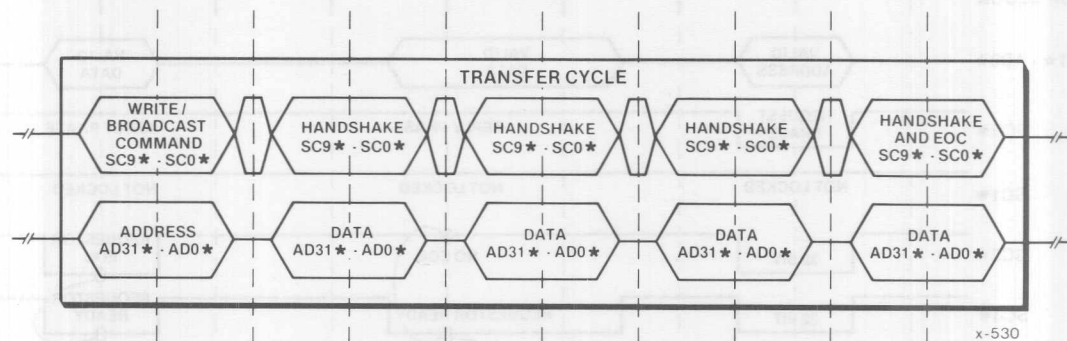


Figure 3-14. Block Diagram Of Broadcast Operation

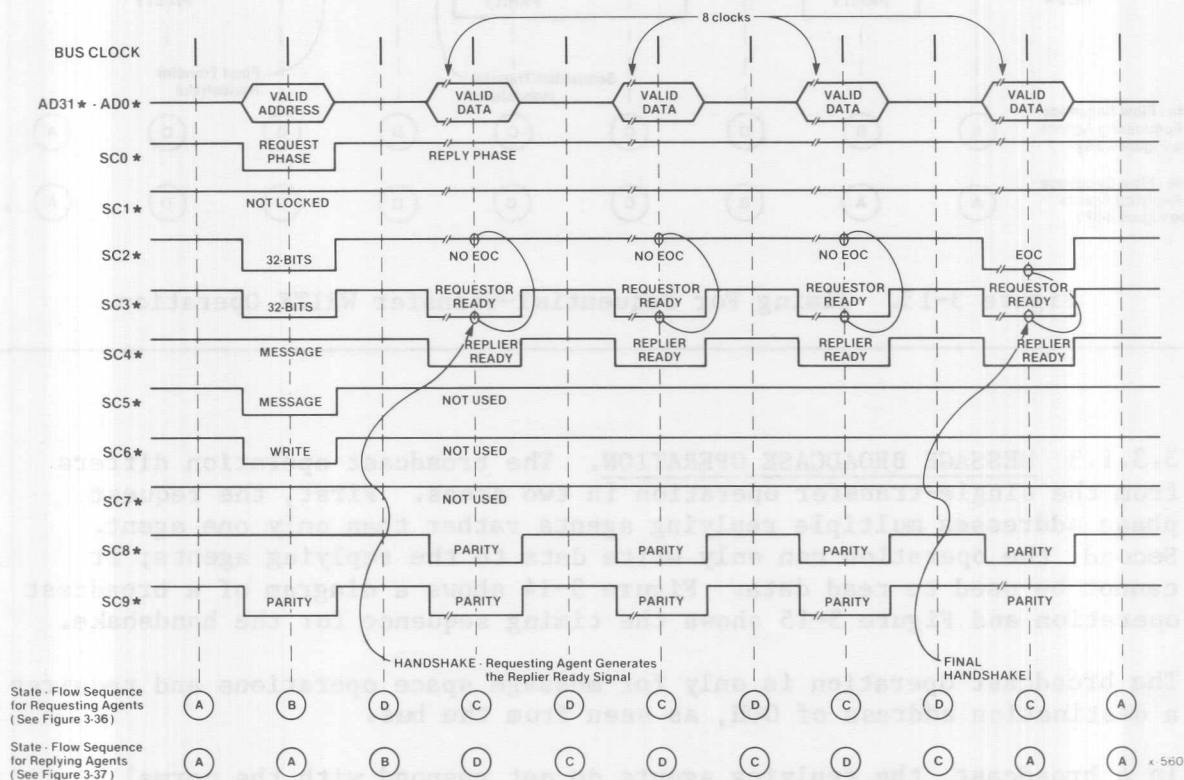


Figure 3-15. Timing For Broadcast Operation

## PARALLEL SYSTEM BUS SPECIFICATION

### 3.3.3 Transfer Cycle LOCK Operation

By asserting  $SC1^*$ , the bus owner may ignore any pending requests and LOCK the bus. If the bus owner locks the bus, then all agents in the resolution phase remain in that phase until the bus owner unlocks the bus. When  $SC1^*$  is held active, all accessed multi-ported resources on the Parallel System bus are locked and remain locked until  $SC1^*$  is removed. Figure 3-16 shows the timing sequence for an agent locking the bus.

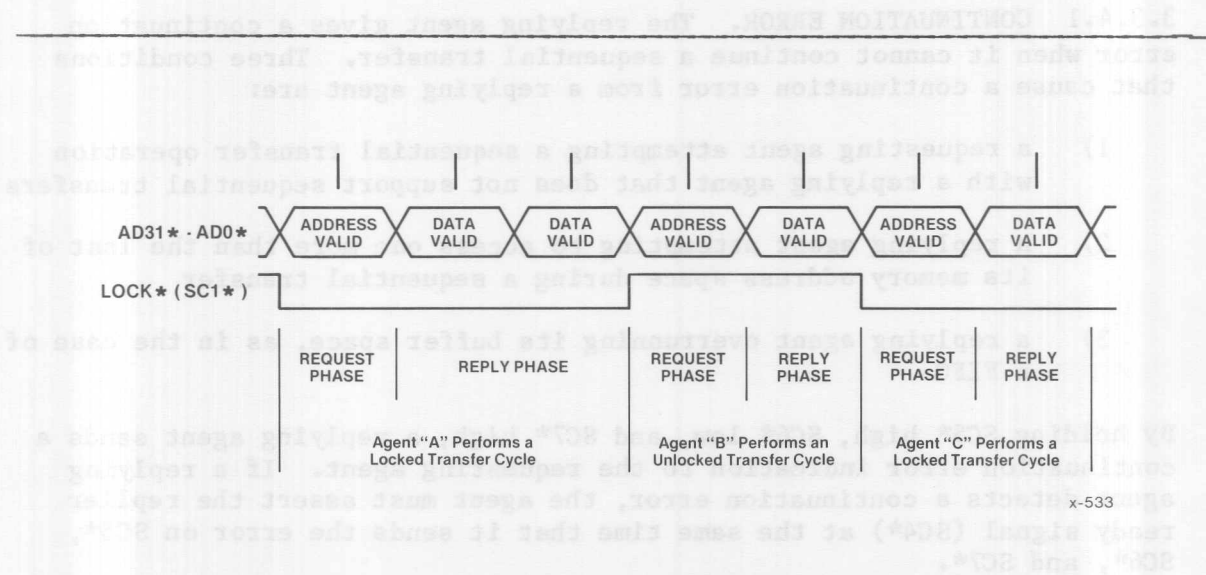


Figure 3-16. Timing Sequence For  $LOCK^*$  Signal Operation

### 3.3.4 Errors In Transfer Cycles

During transfer cycles, the requesting agents continually check for five types of agent error: the continuation error, the transfer-width error, the NACK, the agent data error, and the transfer-not-understood error.

#### NOTE

Agent errors are not to be confused with exceptions that cause an exception cycle. Agent errors do not cause exception cycles and do not terminate arbitration cycles, however, they do terminate the transfer cycle in which they are signaled.



## PARALLEL SYSTEM BUS SPECIFICATION

The agent error is the result of physical problems in the transfer cycle and local to the replying agent. Typically, the problem can be identified and corrected by the requesting agent. On detecting an agent error, a replying agent returns an error code on SC5\*-SC7\* to the requesting agent during the reply phase of the transfer cycle. All types of agent errors terminate the transfer cycle in which they are sensed. Each agent error is described in the following paragraphs.

**3.3.4.1 CONTINUATION ERROR.** The replying agent gives a continuation error when it cannot continue a sequential transfer. Three conditions that cause a continuation error from a replying agent are:

- 1) a requesting agent attempting a sequential transfer operation with a replying agent that does not support sequential transfers
- 2) a replying agent attempting to access one more than the last of its memory address space during a sequential transfer
- 3) a replying agent overrunning its buffer space, as in the case of a FIFO

By holding SC5\* high, SC6\* low, and SC7\* high, a replying agent sends a continuation error indication to the requesting agent. If a replying agent detects a continuation error, the agent must assert the replier ready signal (SC4\*) at the same time that it sends the error on SC5\*, SC6\*, and SC7\*.

**3.3.4.2 TRANSFER-WIDTH ERROR.** The replying agent gives a transfer-width error when it cannot perform an operation as wide as requested.

The replying agent receives a data width code from the requesting agent during the request phase of a transfer cycle. By comparing the request with its actual interface width, the replying agent determines whether or not it can perform the operation.

The replying agent generates a transfer-width error for all operations where the requested width for an operation is greater than the width of the interface on the replying agent. If the requesting agent attempts to transfer a data width less than that required by the replying agent, the replying agent does not generate a transfer-width error for the operation.

By holding SC5\* low, SC6\* high, and SC7\* high, an agent sends a transfer-width error indication to the requesting agent. If a replying agent detects a transfer-width error, the agent must assert the replier ready signal (SC4\*) at the same time that it sends the error on SC5\*, SC6\* and SC7\*.

PARALLEL SYSTEM BUS SPECIFICATION

3.3.4.3 MESSAGE ERROR. The replying agent sends a negative acknowledge (NACK) as a response to a message operation that it cannot perform. A replying agent generates the NACK only when the operation involves the message space. On sensing a NACK, a requesting agent may retry the transfer depending on the implementation. By holding SC5\* high, SC6\* high, and SC7\* low, the replying agent sends a negative acknowledge (NACK) response.

3.3.4.4 DATA ERROR. The an agent sends a data error when it senses an integrity problem with its local data. By holding SC5\* low, SC6\* high, and SC7\* low, the agent sends a data error indication.

3.3.4.5 TRANSFER-NOT-UNDERSTOOD ERROR. This occurs as a result of detecting multiple errors, such as a width error plus a continuation error, or when a replying agent cannot perform the operation requested in a transfer. By holding SC5\* low, SC6\* low, and SC7\* high, a replying agent sends a transfer-cycle-not-understood error to the requesting agent.

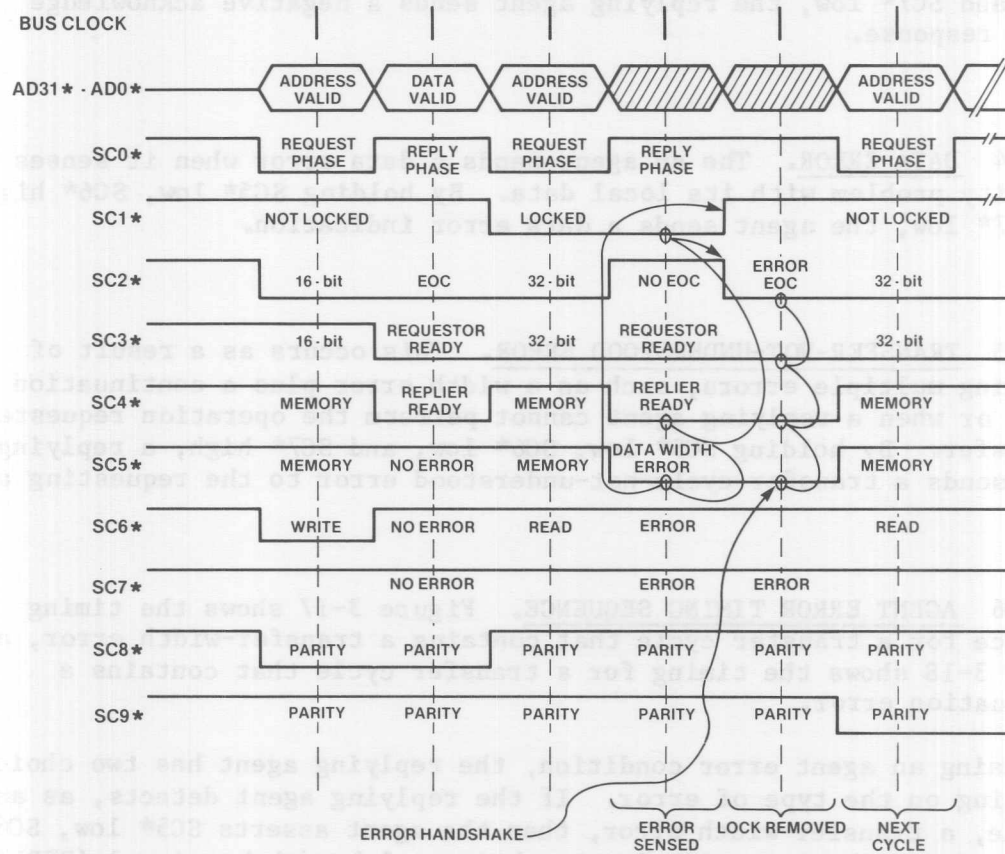
3.3.4.6 AGENT ERROR TIMING SEQUENCE. Figure 3-17 shows the timing sequence for a transfer cycle that contains a transfer-width error, and Figure 3-18 shows the timing for a transfer cycle that contains a continuation error.

On sensing an agent error condition, the replying agent has two choices, depending on the type of error. If the replying agent detects, as an example, a transfer-width error, then the agent asserts SC5\* low, SC6\* high, and SC7\* high in addition to the normal handshake signal (REPLIER READY) on SC4\* during the reply phase. If the replying agent detects a continuation error, the agent asserts SC5\* high, SC6\* low, and SC7\* high in addition to the normal handshake during the reply phase.

When the requesting agent receives an error indication during a handshake, the error forces the requesting agent to assert an END-OF-CYCLE signal on SC2\* during the next clock cycle.

All operations within the replying agent stop until the requesting agent sends the END-OF-CYCLE signal onto the bus.

# PARALLEL SYSTEM BUS SPECIFICATION



x-543

Figure 3-17. Timing Sequence For Transfer Width Agent Error

# PARALLEL SYSTEM BUS SPECIFICATION

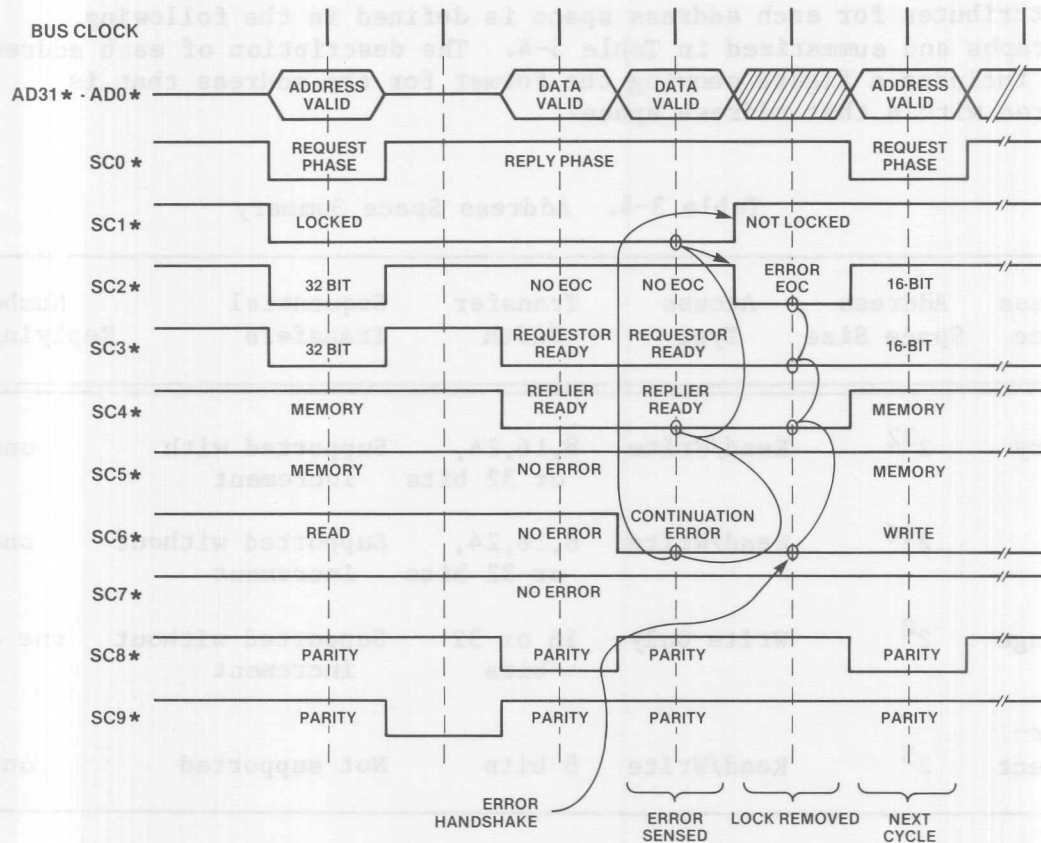


Figure 3-18. Timing Sequence For Continuation Agent Error

## 3.3.5 Address Space Definitions In Transfer Cycles

During the request phase, the requesting agent uses the SC4\* and SC5\* signals to select one of four independent address spaces on the Parallel System bus:

SC5*	SC4*	Address Space Selected
H	H	Memory space
H	L	I/O space
L	H	Message space
L	L	Interconnect space



## PARALLEL SYSTEM BUS SPECIFICATION

Each of the four address spaces has different characteristics, capabilities, and uses in the Multibus II bus architecture. When an agent performs a transfer cycle involving one of the address spaces, that agent is responsible for adhering to the protocol governing access to and use of that address space.

The attributes for each address space is defined in the following paragraphs and summarized in Table 3-4. The description of each address space includes a figure showing the format for the address that is required within that address space.

Table 3-4. Address Space Summary

Address Space	Address Space Size	Access Type	Transfer Width	Sequential Transfers	Number of Replying Agents
Memory	2 <sup>32</sup>	Read/Write	8,16,24, or 32 bits	Supported with increment	one
I/O	2 <sup>16</sup>	Read/Write	8,16,24, or 32 bits	Supported without increment	one
Message	2 <sup>8</sup>	Write Only	16 or 32 bits	Supported without increment	one or more
Inter-connect	2 <sup>5</sup>	Read/Write	8 bits	Not supported	one

**3.3.5.1 MEMORY SPACE ACCESS PROTOCOL.** The memory space defines the memory available on the Parallel System bus. The Multibus II bus architecture defines the use of the memory space with specific protocol, as follows:

- 1) The data width (during the reply phase of the transfer cycle) for a memory space operation must be either eight, sixteen, twenty-four, or thirty-two bits.
- 2) The address width (during the request phase of the transfer cycle) must always be thirty-two bits for an access to memory space. See Figure 3-19.
- 3) Requesting agents that access memory space must receive a response from only one replying agent.
- 4) The replying agent must increment the initial address given by the requesting agent to obtain the address for subsequent accesses of data when performing a transfer cycle that requires sequential accesses of memory.

# PARALLEL SYSTEM BUS SPECIFICATION

- 5) For sequential-access operations, the address incrementing algorithm varies depending on the data width that is required by the requesting agent. For an 8-bit transfer, the address is incremented by one at each access; for a 16-bit transfer, the address is incremented by two at each access, and so on. Refer to Figure 3-20.
- 6) The Multibus II bus architecture does not support agents that perform sequential memory accesses at the twenty-four bit data width or 16-bit data width not aligned on 16-bit boundaries.

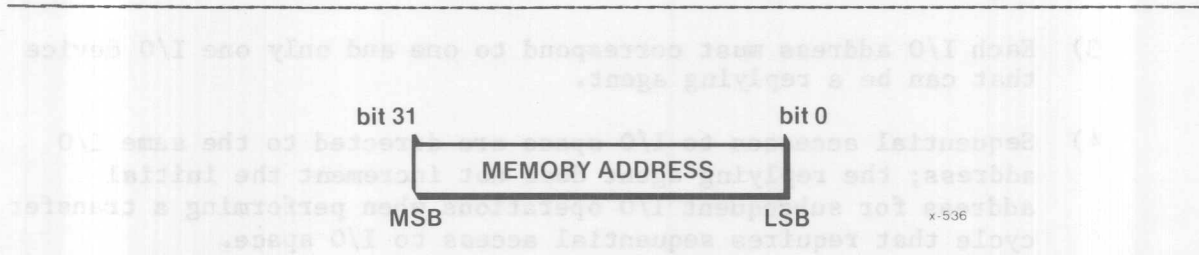


Figure 3-19. Address Format For Operations Using Memory Address Space

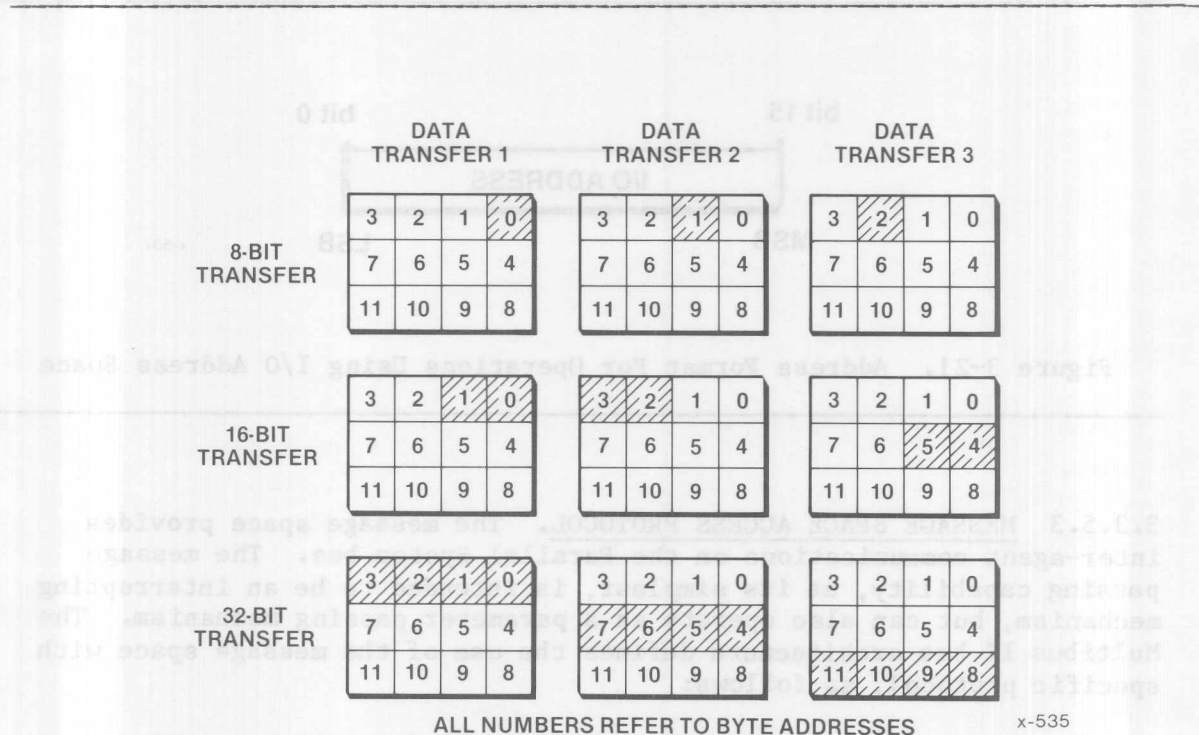


Figure 3-20. Block Diagram Of Sequential Data Transfers In Memory Space

## PARALLEL SYSTEM BUS SPECIFICATION

**3.3.5.2 I/O SPACE ACCESS PROTOCOL.** The I/O space defines the I/O devices that are available on the Parallel System bus. The Multibus II bus architecture defines the use of the I/O space on the Parallel System bus with specific protocol, as follows:

- 1) The data width (during the reply phase of the transfer cycle) for an I/O space operation must be either eight, sixteen, twenty-four, or thirty-two bits.
- 2) The address width during the request phase of the transfer cycle must always be sixteen bits for an access to I/O space. See Figure 3-21.
- 3) Each I/O address must correspond to one and only one I/O device that can be a replying agent.
- 4) Sequential accesses to I/O space are directed to the same I/O address; the replying agent does not increment the initial address for subsequent I/O operations when performing a transfer cycle that requires sequential access to I/O space.
- 5) The Multibus II bus architecture does not support agents that perform sequential I/O accesses at the twenty-four bit data width or at the non-aligned 16-bit data width.

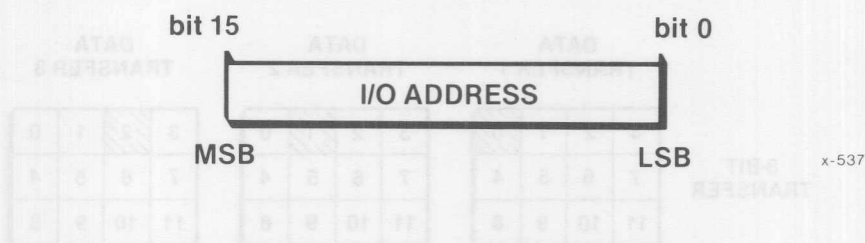


Figure 3-21. Address Format For Operations Using I/O Address Space

**3.3.5.3 MESSAGE SPACE ACCESS PROTOCOL.** The message space provides inter-agent communications on the Parallel System bus. The message passing capability, at its simplest, is intended to be an interrupting mechanism, but can also operate as a parameter passing mechanism. The Multibus II bus architecture defines the use of the message space with specific protocol, as follows:

- 1) The data transfer is one-directional, from the requesting agent to the replying agent(s).
- 2) Only the message header is defined in this document.

## PARALLEL SYSTEM BUS SPECIFICATION

- 3) The data width for a message space operation must be either sixteen or thirty-two bits.
- 4) The address width (during the request phase of the transfer cycle) must always be sixteen bits for an access to message space. See Figure 3-22.
- 5) The message space supports agents that perform broadcast operations to multiple replying agents.
- 6) The message space supports sequential access operations.
- 7) Sequential accesses to message space are directed to the same message address; the replying agent does not increment the initial address for subsequent message space operations when performing a transfer cycle that requires sequential accesses.
- 8) The Multibus II bus architecture supports two types of message space commands: unsolicited and solicited. Unsolicited messages are unexpected, bounded, interrupt messages that move small amounts of data or parameters. Solicited messages are negotiable, data messages that move blocks of data. Additional details are contained in the System Interface Specification.

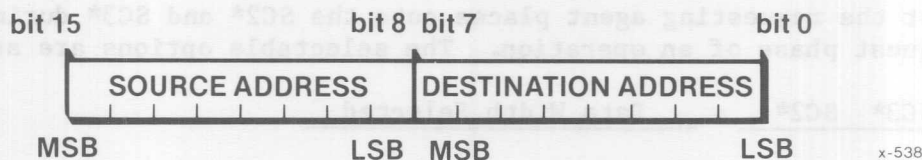


Figure 3-22. Address Format For Operations Using Message Space

**3.3.5.4 INTERCONNECT SPACE ACCESS PROTOCOL.** The interconnect space provides configuration information for the resources on the Parallel System bus. The interconnect space is intended to be a system initialization mechanism, but can also operate as a parameter passing mechanism. The Multibus II bus architecture defines the use of the message space with specific protocol, as follows:

- 1) The data transfer is bi-directional (width is 8-bits only.)
- 2) Each agent in the iPSB bus must have its own unique cardslot address, assigned by the CSM.



## PARALLEL SYSTEM BUS SPECIFICATION

- 3) The address width (during the request phase of the transfer cycle) must always be sixteen bits wide, with 9 bits of offset register address and 5 bits of board identification address. See Figure 3-23.
- 4) The protocol does not support sequential access or broadcast operations in the interconnect space.

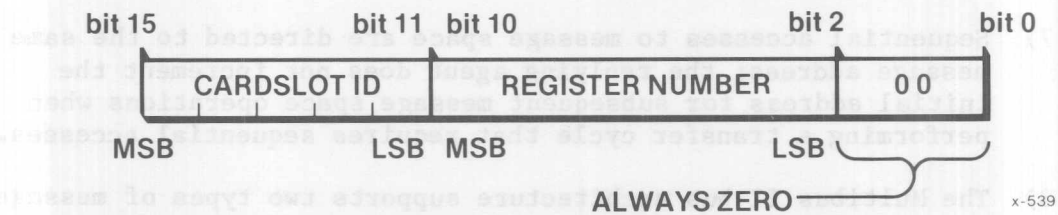


Figure 3-23. Address Format For Operations Using Interconnect Space

### 3.3.6 Data Width During Transfer Cycles

The data width for an operation is defined via the data-width parameter that the requesting agent places onto the SC2\* and SC3\* during the request phase of an operation. The selectable options are as follows:

SC3*	SC2*	Data Width Selected
H	H	8-bit
H	L	16-bit
L	H	24-bit
L	L	32-bit

When requesting agents perform an operation on the Parallel System bus, the agents are allowed several data alignment options depending on the nature of the agents and the operation. Data alignment requirements for the four address spaces are described in two sections; alignment for agents doing memory, interconnect, and I/O space operations and alignment for agents doing message space operations.

#### 3.3.6.1 DATA ALIGNMENT IN MEMORY, I/O, AND INTERCONNECT ADDRESS SPACE.

The protocol allows the use of seven data alignment options for requesting and replying agents that use the memory, I/O, and Interconnect address space.

## PARALLEL SYSTEM BUS SPECIFICATION

The requesting agent controls the alignment of the interface by controlling the condition of address bits 0 and 1 via address lines AD0\* and AD1\* and controls the data width via the SC2\* and SC3\* lines on the interface.

Table 3-5 lists the alignment options for the interface. In Table 3-5, the columns represent the four bytes of the 32-bit address/data bus on the Parallel System bus; BYTE 0 is the least significant byte (AD7\* through AD0\*) and BYTE 3 is the most significant (AD31\* through AD24\*).

Table 3-5. Data Alignments for Operations Using Memory, I/O,  
And Interconnect Address Space

Transfer Width	Request Phase Bits				Reply Phase Byte Alignments			
	Width		Address		AD31*-	AD23*-	AD15*-	AD7*-
	SC3*	SC2*	AD1*	AD0*	AD24*	AD16*	AD8*	AD0*
8-bits (Notes 1,5)	H	H	X	H	I	I	I	D
8-bits (Note 1)	H	H	X	L	I	I	D	I
16-bits (Note 1)	H	L	X	H	I	I	D	D
16-bits (Note 4)	H	L	H	L	I	D	D	I
24-bits (Note 4)	L	H	H	H	I	D	D	D
24-bits (Note 4)	L	H	H	L	D	D	D	I
32-bits (Notes 1,4)	L	L	H	H	D	D	D	D

Notes: 1. Identifies those alignments allowed with sequential transfer operations.

2. Abbreviations

D = Valid data on the bus.

I = Invalid portion of bus; must be ignored by replying agent during write operations, but can be driven by requesting agent.

X = Either high or low logic level is acceptable.

L = Low (false) logic level required.

H = High (true) logic level required.

3. All unlisted configurations are not supported in the protocol and reported as transfer-not-understood errors.

4. Requires a 32-bit agent.

5. The only allowed data alignment for Interconnect space operations.

The alignment restrictions for sequential transfers are as follows:

- 1) For a sequential 8-bit operation, the low address bit may be either 0 or 1, depending on which byte is transferred, and the data is transferred on either Byte 0 or Byte 1.

## PARALLEL SYSTEM BUS SPECIFICATION

- 2) For a sequential 16-bit operation, the low address bit must always be 0 and the data is transferred on Byte 0 and Byte 1.
- 3) No sequential 24-bit operations or 24-bit agents are supported.
- 4) For a sequential 32-bit operation, the least significant two bits of address must always be 00 and the data is transferred on Byte 0, Byte 1, Byte 2 and Byte 3.

3.3.6.2 DATA ALIGNMENT IN MESSAGE SPACE. Table 3-6 shows the data alignments allowed for message space operations. The alignment restrictions for message space differ from those presented for the memory, I/O, and interconnect space for two reasons:

- 1) An agent must know beforehand how much information to accept when it receives a message operation from some other agent on the Parallel System bus.
- 2) Each address in message space identifies a logical module rather than a physical storage location.

An agent uses message space to transfer either sixteen or thirty-two bit messages or to send/receive interrupts on the iPSB Bus.

Table 3-6. Data Alignment for Message Space Operations

Transfer Width Of Agents	Request Phase Bits		Reply Phase Byte Alignments			
	Width		AD31*-	AD23*-	AD15*-	AD7*-
	SC3*	SC2*	AD24*	AD16*	AD8*	AD0*
16-bits	H	L	I	I	D	D
32-bits	L	L	D	D	D	D

**Notes: Abbreviations**

D = Valid data on the bus.

I = Invalid portion of bus; must be ignored by replying agent during write operations, but can be driven by requesting agent.

L = Low (false) logic level required.

H = High (true) logic level required.

Figure 3-24 shows the format of a typical message on the Parallel System bus. Each requesting agent creates 16-bit or 32-bit messages containing five fields:

## PARALLEL SYSTEM BUS SPECIFICATION

- 1) address field identifying the requesting agent
- 2) address field identifying the destination agent
- 3) a field providing command specific information
- 4) a field containing the command
- 5) optional field(s) containing either 16-bits or 32-bits of data

Each field of the message space operation is described in the following paragraphs.

The source and destination address fields contain an 8-bit identifier that uniquely addresses each agent for message operations on the Parallel System bus. The address fields contain 8-bit identifier codes for each of the agents involved in the message passing operation, except for a broadcast operation. The source address field identifies the agent that sources the message, and the destination address field identifies the agent that receives the message.

The type field provides a description of the operation that the agents are to perform. The type specific field changes functions depending on the type. Refer to the System Interface Specification.

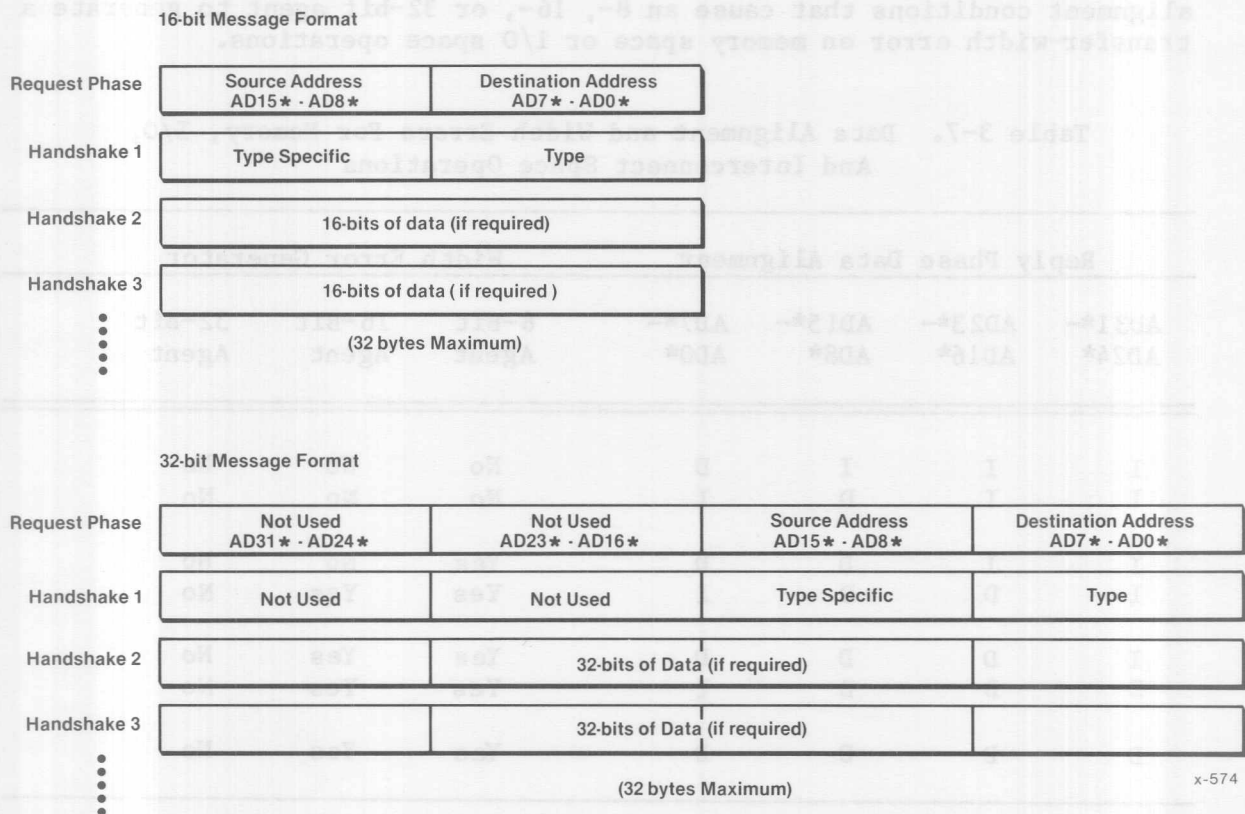


Figure 3-24. 16-Bit And 32-Bit Message Format Examples



## PARALLEL SYSTEM BUS SPECIFICATION

The data fields contain the message information. The information may be either 16-bits or 32-bits wide, depending on the agents involved in the transfer.

On sending a message, a requesting agent transfers the fields onto AD31\* through AD0\* for the replying agent during a transfer cycle. Depending on the physical properties of the agent, the typical message consists of a sequential access in which the requesting agent performs one request phase and extends the reply phase into one to thirty-one handshake cycles by withholding the end-of-cycle handshake. The replying agent reports any problems via the agent error lines.

Refer to the System Interface Specification portion of the Multibus II Bus Architecture Specification for additional information on message space.

Refer to the section on Interrupts for more information on interrupts on the Multibus II bus architecture.

### 3.3.7 Data Alignment Error Reporting During Transfer Cycles

All agents may check for transfer-width errors, including interface width mismatch and data misalignment problems. Table 3-7 shows the data alignment conditions that cause an 8-, 16-, or 32-bit agent to generate a transfer-width error on memory space or I/O space operations.

Table 3-7. Data Alignment and Width Errors For Memory, I/O,  
And Interconnect Space Operations

Reply Phase Data Alignment				Width Error Generator		
AD31*- AD24*	AD23*- AD16*	AD15*- AD8*	AD7*- AD0*	8-Bit Agent	16-Bit Agent	32-Bit Agent
I	I	I	D	No	No	No
I	I	D	I	No	No	No
I	I	D	D	Yes	No	No
I	D	D	I	Yes	Yes	No
I	D	D	D	Yes	Yes	No
D	D	D	I	Yes	Yes	No
D	D	D	D	Yes	Yes	No

Notes: D = Active, contains valid data.

I = Ignored by replying agent, may be driven.

Interconnect agents are 8-bit agents (only).

## PARALLEL SYSTEM BUS SPECIFICATION

Table 3-8 shows the data alignment conditions that cause a 16-bit or a 32-bit agent to generate a transfer-width error during a message space operation on the Parallel System bus.

As Table 3-7 shows, some replying agents return transfer-width errors on certain legal operations. As an example, when a 16-bit requesting agent performs a 16-bit transfer to an 8-bit replying agent, the replying agent returns a transfer-width error.

Table 3-8. Data Alignment And Width Errors For Message Space

Reply Phase Data Alignment				Width Error Generator	
AD31*- AD24*	AD23*- AD16*	AD15*- AD8*	AD7*- AD0*	16-bit Agent	32-bit Agent
I	I	D	D	No	No
D	D	D	D	Yes	No

### 3.3.8 Data Alignment Interface Example

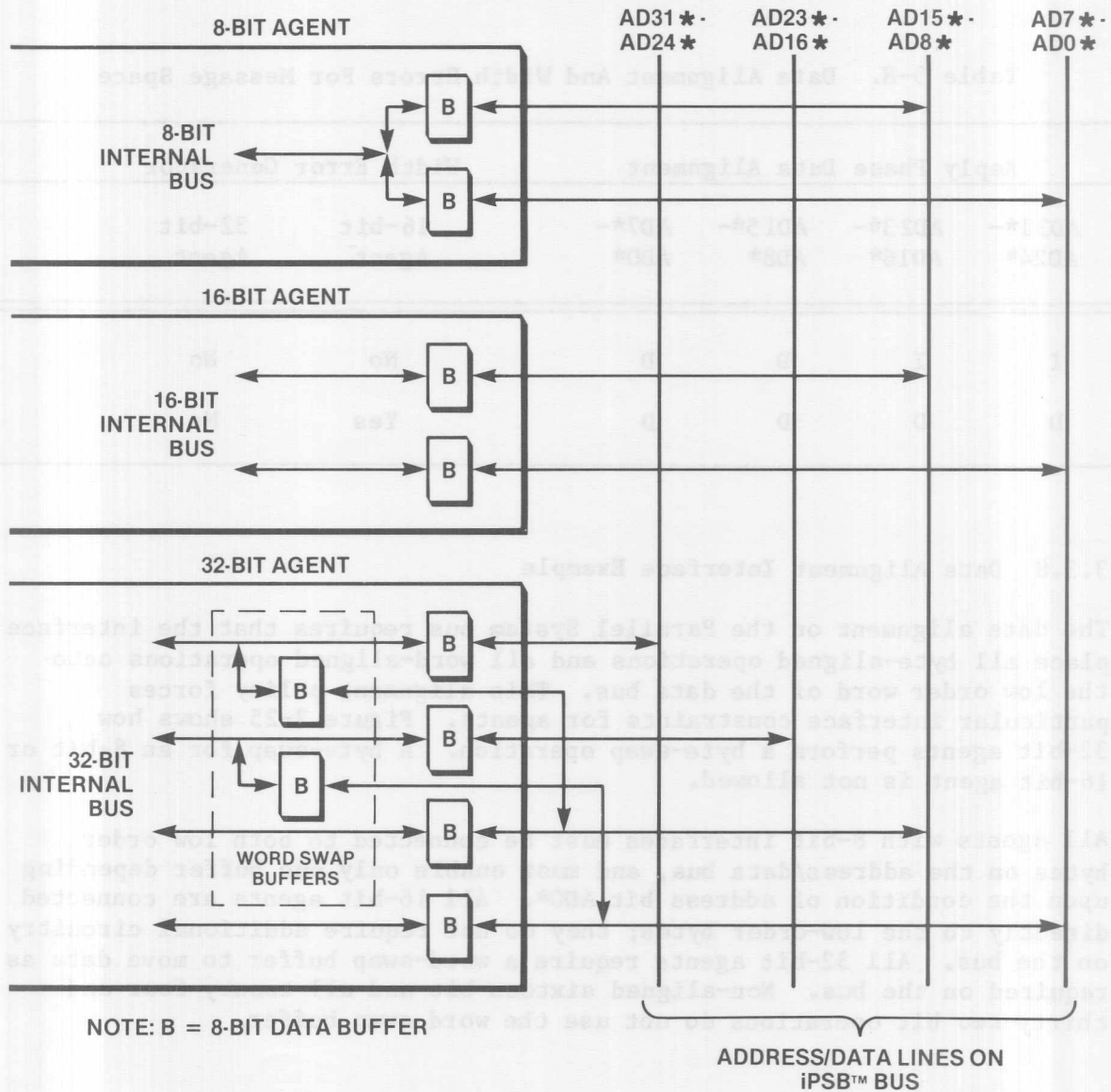
The data alignment on the Parallel System bus requires that the interface place all byte-aligned operations and all word-aligned operations onto the low order word of the data bus. This alignment policy forces particular interface constraints for agents. Figure 3-25 shows how 32-bit agents perform a byte-swap operation. A byte-swap for an 8-bit or 16-bit agent is not allowed.

All agents with 8-bit interfaces must be connected to both low order bytes on the address/data bus, and must enable only one buffer depending upon the condition of address bit AD0\*. All 16-bit agents are connected directly to the low-order bytes; they do not require additional circuitry on the bus. All 32-bit agents require a word-swap buffer to move data as required on the bus. Non-aligned sixteen bit and all twenty-four and thirty-two bit operations do not use the word swap buffer.

### 3.3.9 Interrupts

The Multibus II bus architecture supports interrupts on the iPSB Bus through use of the messages. When an agent determines that it must send an interrupt message onto the iPSB bus, the agent requests access to the bus, performs a transfer cycle that sends an unsolicited message, and releases the bus.

# PARALLEL SYSTEM BUS SPECIFICATION



x-548

Figure 3-25. Interface Requirements For Data Alignment

## PARALLEL SYSTEM BUS SPECIFICATION

Figure 3-26 shows how the 16-bit or 32-bit interrupt message is encoded on the bus. An Interrupt message is an unsolicited message with a null data field. As the (interrupt) requesting agent performs the first part of the operation, it transfers an 8-bit interrupt source ID (the ID of the agent sourcing the interrupt request), and an 8-bit interrupt destination ID (the ID of the agent that must service the interrupt request) onto the bus. In the second part of the operation, the (interrupt) requesting agent transfers an 8-bit reserved field (should be ignored) and an 8-bit interrupt type ID. Refer to the System Interface Specification for more information on type field details.

All agents receive the message information. Only the replying agent (the agent selected by the destination ID to service the interrupt) responds to the message by interrupting its on-board processor and beginning an interrupt servicing routine. The type of service may depend on the interrupt type.

In Figure 3-27, a simple requesting agent requires service. The agent then forms an interrupt message with replying agent's address and puts the message on the bus. The replying agent converts the message into a local interrupt and initiates an interrupt service routine. The local replying agent then responds with service.

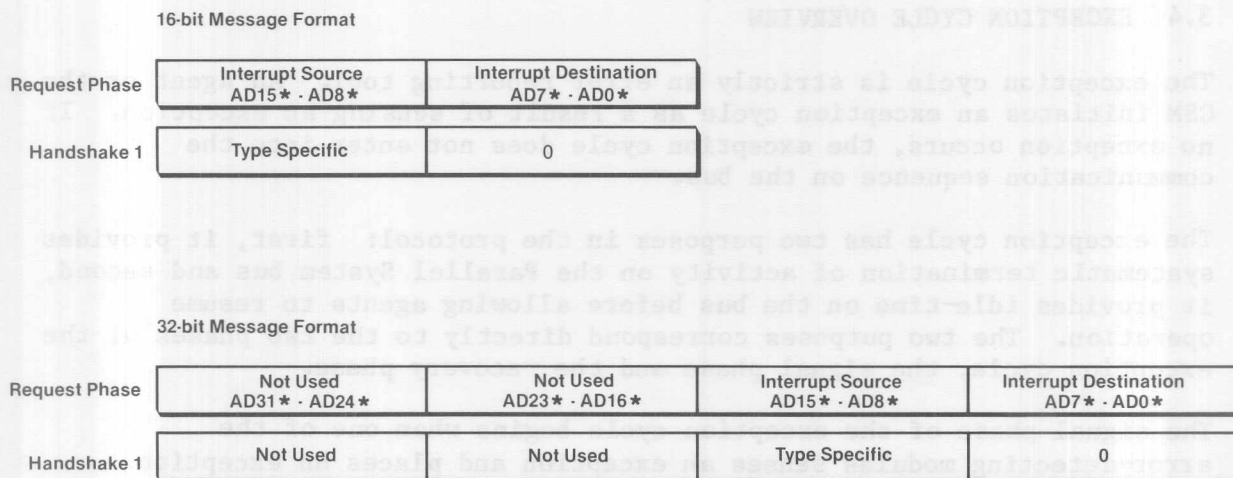


Figure 3-26. Interrupt Message Format



## PARALLEL SYSTEM BUS SPECIFICATION

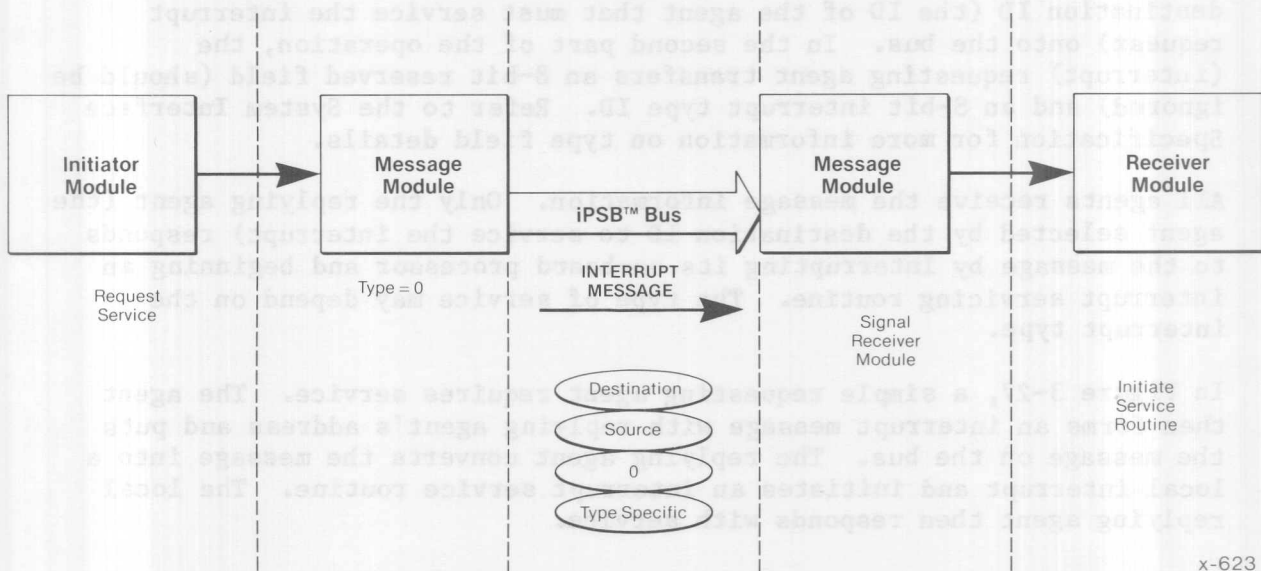


Figure 3-27. Interrupt Message Sequence

### 3.4 EXCEPTION CYCLE OVERVIEW

The exception cycle is strictly an error reporting tool. An agent or the CSM initiates an exception cycle as a result of sensing an exception. If no exception occurs, the exception cycle does not enter into the communication sequence on the bus.

The exception cycle has two purposes in the protocol: first, it provides systematic termination of activity on the Parallel System bus and second, it provides idle-time on the bus before allowing agents to resume operation. The two purposes correspond directly to the two phases of the exception cycle, the signal phase and the recovery phase.

The signal phase of the exception cycle begins when one of the error-detecting modules senses an exception and places an exception indication onto the bus. On receiving an exception indication, the requesting agent aborts any transfer cycles and inhibits them until the exception cycle is completed. An agent could be operating at any point in either an arbitration cycle or a transfer cycle when an exception terminates that cycle and starts an exception cycle. The net effect of the exception cycle is to terminate all bus activity and to hold the bus idle for a set amount of time. The signal phase continues until the error-detecting module deactivates the exception on the bus.

## PARALLEL SYSTEM BUS SPECIFICATION

The recovery phase of the exception cycle begins after the exception signals become inactive. The recovery phase is a fixed-duration delay (in terms of bus cycles) that allows time for the Parallel System bus signals to settle before starting more transfer cycles.

Figure 3-28 shows the operation of the exception cycle signals and their relationship to the phases of the exception cycle.

When an agent sends an exception the agent holds one of the exception signals (BUSERR\* or TIMEOUT\*) active for a minimum of one clock cycle. The agent may hold the exception signal active on the bus for any duration, however, doing so extends the signal phase of the cycle. On deactivating the signal, the agent places the protocol into the recovery phase of the exception cycle. The recovery phase is a pre-determined number of clock cycles in duration.

### 3.4.1 Causes of Exception Cycles

Exception cycles are caused when an agent senses an active exception signal. The Parallel System bus provides facilities for agents to sense and report two types of exception: the timeout exception and the bus error exception. Any agent (and in one case the CSM) can detect these exceptions and start an exception cycle at any time on the Parallel System bus. Each of the exceptions serves a different purpose.

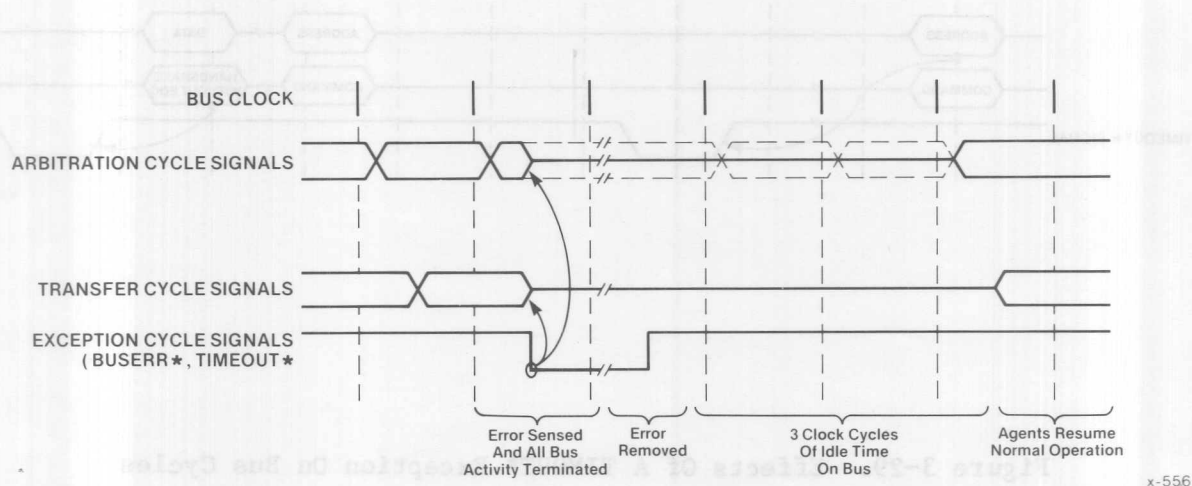


Figure 3-28. Exception Cycle Signal Relationships

## PARALLEL SYSTEM BUS SPECIFICATION

**3.4.1.1 TIMEOUT EXCEPTION.** The CSM sends a timeout exception onto the Parallel System bus when the current transfer cycle exceeds one millisecond. The time limit configured in the CSM is the same for all types of transfer cycles and the same for operations to all address spaces. The Parallel System bus provides a dedicated line (TIMOUT\*) on the bus for passing the timeout exception among all agents. All agents must continually be aware of the condition of the timeout exception signal, TIMOUT\*.

The timeout exception is a result of too much time between consecutive handshake signals during a transfer cycle. On sensing a timeout, the CSM activates the TIMOUT\* signal. TIMOUT\* terminates both the arbitration cycle and the transfer cycle in all agents, and begins an exception cycle.

Figure 3-29 shows the timing for a TIMOUT\* exception. As the figure shows, an agent can generate a timeout as a result of a faulty handshake/data transfer during the reply phase or as the result of a faulty address/command transfer during the request phase.

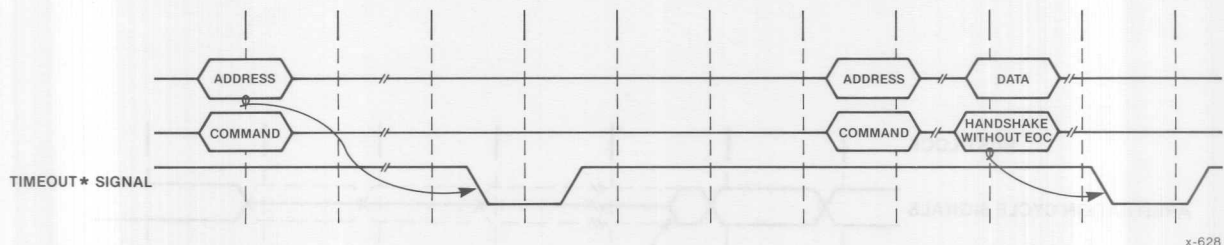


Figure 3-29. Effects Of A TIMOUT\* Exception On Bus Cycles

## PARALLEL SYSTEM BUS SPECIFICATION

**3.4.1.2 BUS ERROR EXCEPTION.** An agent sends a bus error exception whenever it determines that the information on the address/data bus (AD31\* through ADO\*) and system control lines (SC9\*-SC0\*) is in error. Bus error exceptions are parity errors sensed during a data transfer. The Parallel System bus provides a dedicated line for passing the bus error exception (BUSERR\*) signal among agents. All agents must continually be aware of the condition of the bus error exception signal.

Figure 3-30 shows the timing sequence for two cases of detection of a bus error exception. The first BUSERR\* exception occurs after the request phase and terminates the transfer cycle during the request phase. The second terminates the transfer cycle during the reply phase. An agent asserts BUSERR\* one clock cycle after it detects the exception. One clock cycle later, the exception cycle terminates both transfer cycles and arbitration cycles.

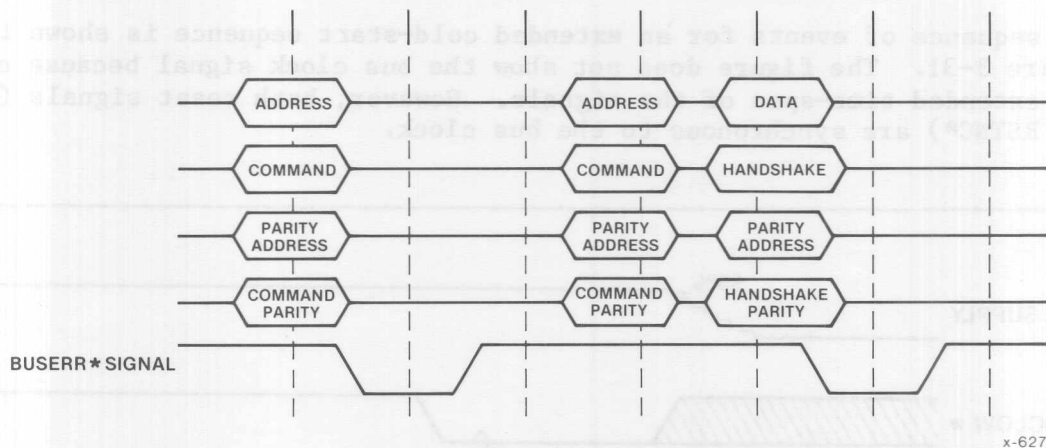


Figure 3-30. Effects Of A BUSERR\* Exception On Bus Cycles

## 3.5 CENTRAL CONTROL FUNCTIONS

The Parallel System bus provides three system-level functions via the central control signal group. Those functions include a power-up sequence control (cold-start), initialization sequence control (warm-start), and powerfail-recovery control. Each is described in the following paragraphs.



## PARALLEL SYSTEM BUS SPECIFICATION

### 3.5.1 Power-Up Function (Cold-Start)

The power-up reset, sometimes referred to as a "cold-start" reset, initializes all agents in the system. The cold-start period provides two benefits in the system: it ensures a uniform initialization period for all agents on application of power, and it gives all agents in the system the opportunity to begin operation from a known state.

On power-up, the CSM drives the DCLOW\* and RST\* signals active and holds the PROT\* signal inactive within 1 ms. Thereafter, the CSM holds the DCLOW\* signal active for 2.5 ms during the cold-start reset. Circuitry on the CSM holds RST\* active for 50 ms (minimum) after the DCLOW\* signal goes inactive. If one or more agents require additional time to complete initialization operations, those agents may assert RSTNC\* to extend the duration of the initialization period. RSTNC\* inhibits all agents from starting bus cycles until all agents are ready to proceed.

The combination of both DCLOW\* and RST\* active and PROT\* inactive on the bus (from the CSM) identifies when an agent must perform a power-on reset. Any time that both RST\* and DCLOW\* are active and PROT\* is inactive, agents may assume that a power-on initialization is occurring and should perform the power-up reset sequence. Both DCLOW\* and PROT\* from the CSM are asynchronous to the clock.

The sequence of events for an extended cold-start sequence is shown in Figure 3-31. The figure does not show the bus clock signal because of the extended time-span of the signals. However, both reset signals (RST\* and RSTNC\*) are synchronous to the bus clock.

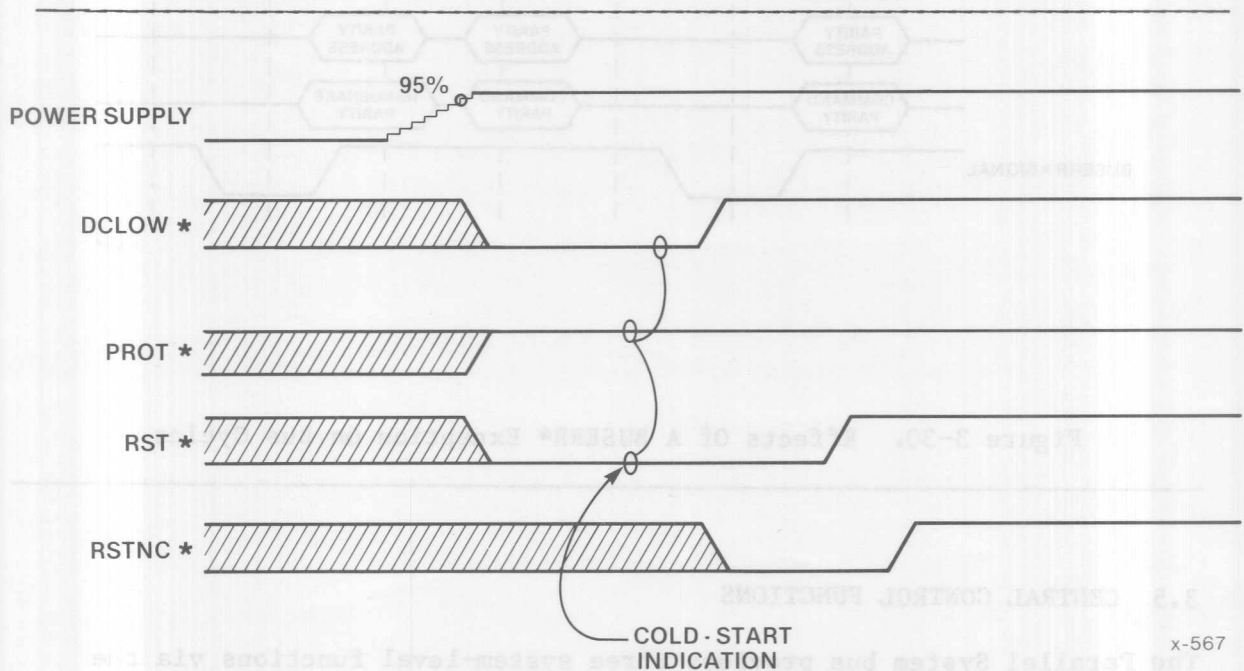


Figure 3-31. Power-On System Reset Sequence

## PARALLEL SYSTEM BUS SPECIFICATION

### 3.5.2 Initialization Sequence (Warm-Start)

The sequence of events for resetting a running system, sometimes referred to as a "warm-start", is shown in Figure 3-32. This type of reset is typically the result of pushing a front panel switch. The figure does not show the bus clock signal because of the extended time-span of the signals. However, both reset signals (RST\* and RSTNC\*) are synchronous to the bus clock.

The CSM causes a warm-start sequence by activating RST\* for a minimum of 50 ms. Agents may extend the initialization period by asserting RSTNC\*. As with the cold-start sequence, RSTNC\* prevents all agents from starting bus cycles until all agents are reset.

Unlike the cold-start, this "warm-start" reset sequence does not assert DCLOW\*. Agents that must differentiate between the two types of reset sequence may do so by examining the condition of the DCLOW\* line on the Parallel System bus during the time that RST\* is active.

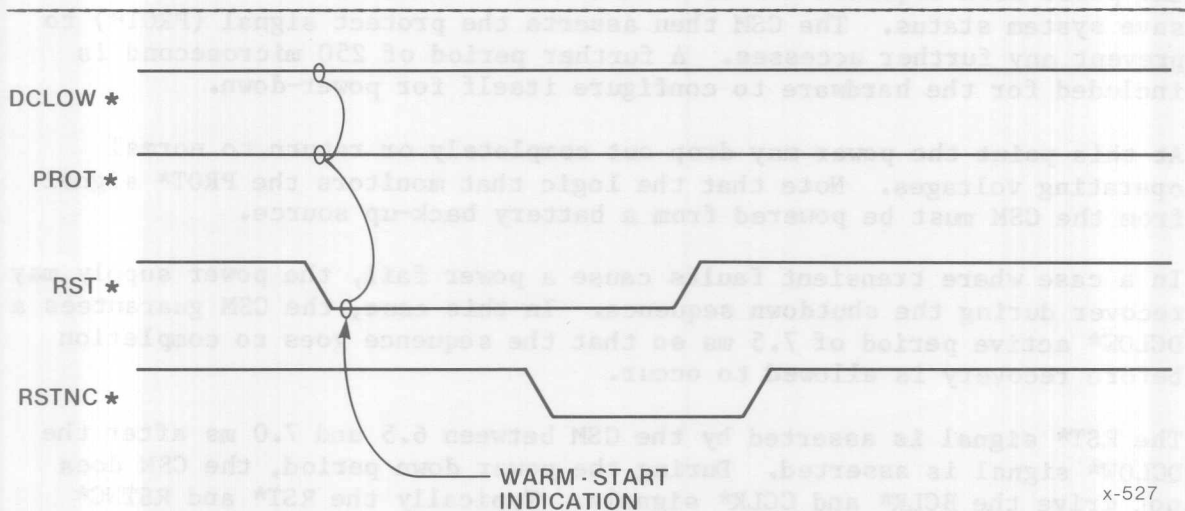


Figure 3-32. Warm-Start Reset Sequence

### 3.5.3 Power Failure And Recovery Sequence

The CSM uses the powerfail and recovery sequence to provide orderly control of system shut-down and start-up during a power failure. The system power supply is expected to provide a power-fail signal to the CSM. When the AC input power drops below an acceptable value, the power supply asserts signals which inform the CSM of the power failure condition. Figure 3-33 shows a power failure and recovery sequence on the Parallel System bus.

This sequence assumes that the system supports battery back-up and allows a reasonable period for the back-up and recovery sequences to execute.

On learning of an imminent power failure (via an early warning mechanism within the power supply), the CSM asserts the DCLOW\* signal onto the Parallel System bus. An active DCLOW\* signal informs all agents of the impending power failure a minimum of 6.5 milliseconds before the power level drops below the minimum level for safe system operation.

The power-down sequence allows from 6.0 to 6.25 ms for the software to save system status. The CSM then asserts the protect signal (PROT\*) to prevent any further accesses. A further period of 250 microsecond is included for the hardware to configure itself for power-down.

At this point the power may drop out completely or return to normal operating voltages. Note that the logic that monitors the PROT\* signal from the CSM must be powered from a battery back-up source.

In a case where transient faults cause a power fail, the power supply may recover during the shutdown sequence. In this case, the CSM guarantees a DCLOW\* active period of 7.5 ms so that the sequence goes to completion before recovery is allowed to occur.

The RST\* signal is asserted by the CSM between 6.5 and 7.0 ms after the DCLOW\* signal is asserted. During the power down period, the CSM does not drive the BCLK\* and CCLK\* signals. Typically the RST\* and RSTNC\* signals are not connected to battery back-up power.

After power returns to its operating level and maintains that level for at least one millisecond, the CSM forces the DCLOW\* signal inactive. About 2.5 milliseconds after removing DCLOW\*, the CSM deactivates the protect signal, PROT\*.

An agent determines whether or not a power failure occurred by examining the PROT\*, DCLOW\*, and RST\* signals. A specific condition of the signals (DCLOW\* inactive, RST\* active, and PROT\* active) provides an indication that a power failure has occurred.

Both DCLOW\* and PROT\* are asynchronous to the bus clock.

## PARALLEL SYSTEM BUS SPECIFICATION

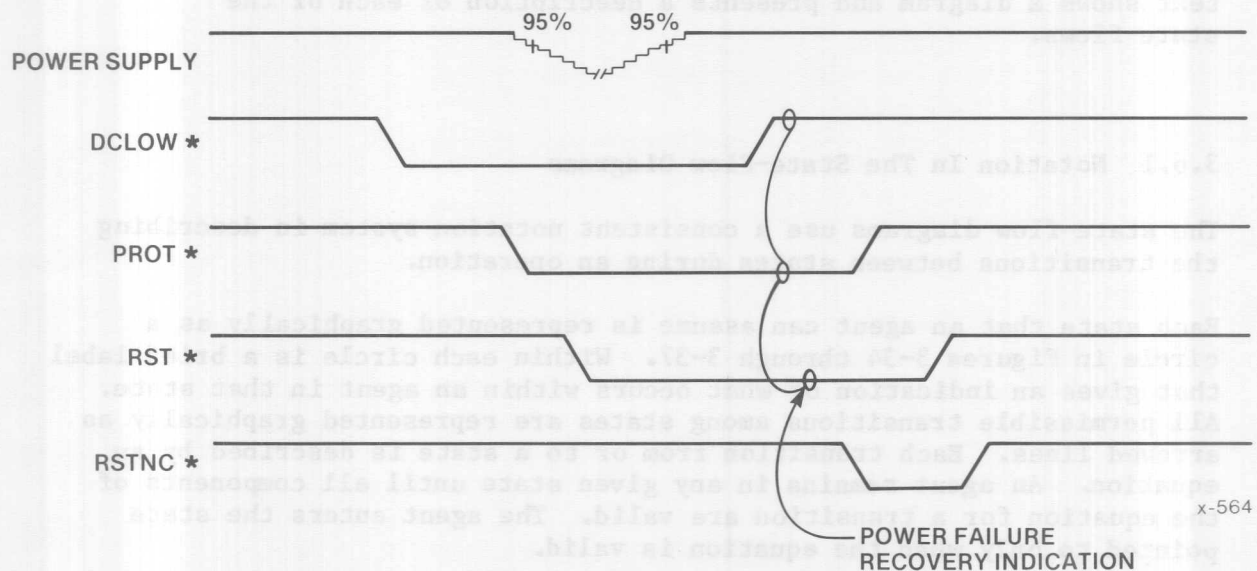


Figure 3-33. Power Failure Recovery Sequence

A power-off sequence is defined for these systems which do not support battery back-up. This occurs when all supply voltages, including the +5 battery, go down. For such systems, DCLOW\* must go active at least 2.5 ms before the power level drops below the safe operating level. PROT\* and RST\* will be held inactive during this period. The CSM may optionally attempt the power-fail sequence until such time as the power actually fails.

Power failure during a cold-start or during a warm-start causes the CSM to not regulate the power shutdown. Either condition should be followed by a cold-start sequence. If a warm-start occurs during a power shutdown, the CSM has the option of either executing the power shutdown and ignoring the reset command, or aborting the shutdown and initiating cold-start on power recovery.

### 3.6 STATE-FLOW DIAGRAMS

This discussion of the Parallel System bus uses state-flow diagrams to describe the operation of agents on the bus. The state-flow diagrams separate the operation of an agent into several steps. The state-flow diagrams for requesting and replying agents are as follows:

- 1) state-flow for agents monitoring the bus
- 2) state-flow for agents in an arbitration cycle
- 3) state-flow for requesting agents in a transfer cycle
- 4) state-flow for replying agents in a transfer cycle



The exception cycle does not have a separate state-flow diagram, however, it causes transitions in each of the four state-flows. The following text shows a diagram and presents a description of each of the state-flows.

### 3.6.1 Notation In The State-Flow Diagrams

The state-flow diagrams use a consistent notation system in describing the transitions between states during an operation.

Each state that an agent can assume is represented graphically as a circle in Figures 3-34 through 3-37. Within each circle is a brief label that gives an indication of what occurs within an agent in that state. All permissible transitions among states are represented graphically as arrowed lines. Each transition from or to a state is described by an equation. An agent remains in any given state until all components of the equation for a transition are valid. The agent enters the state pointed to only when the equation is valid.

The state equations use "AND" to indicate when the AND operator is required and "OR" to indicate when the OR operator is used. Notes are included in some places on the state diagrams and are printed in italics. Square brackets separate equations where two or more independent equations are listed as a cause for a state transition.

In some cases, the equations in the state-flow diagrams contain a function expression in place of a long equation. Where this occurs the functional expression is contained within parenthesis () and defined at the bottom of the diagram.

### 3.6.2 State-Flow Sequence For An Agent Monitoring The Bus

Figure 3-34 presents a state-flow diagram for requesting agents monitoring the condition of the bus. The figure includes two operating states for a requesting agent during bus monitoring, as follows: the NO-TRANSFER-CYCLE state and the TRANSFER-CYCLE state. The following text describes how a requesting agent steps through each state.

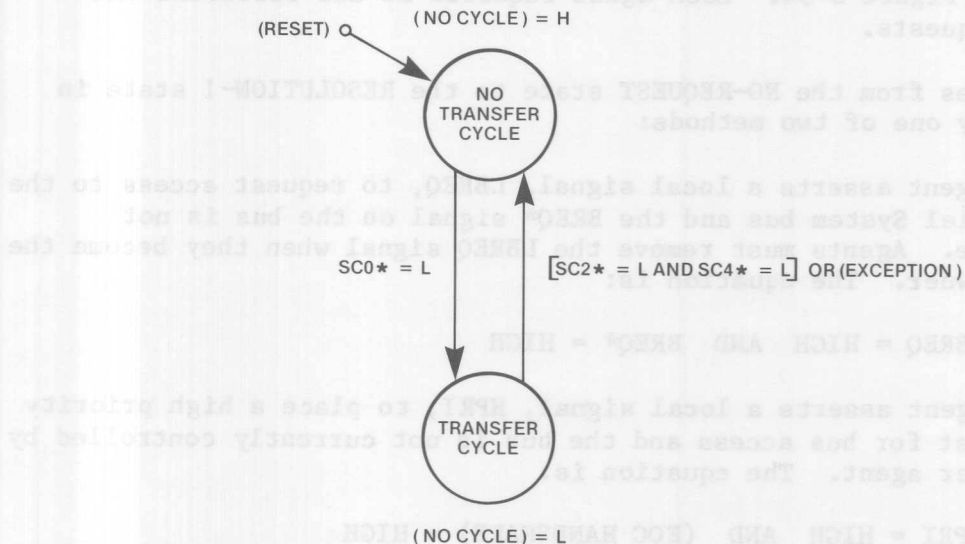
All requesting agents monitor for the NO-CYCLE output of the state machine. Without that indication, agents cannot begin the next resolution phase. When a requesting agent senses the end-of-cycle indication, that agent enters arbitration with all other agents that require access to the bus. All requesting agents monitor the condition of the bus to know when to begin arbitrating for access.

Initially, no agent is performing a transfer cycle and the bus is idle. As a result, all requesting agents are in the NO-TRANSFER-CYCLE state. When one agent starts a transfer cycle, it places a start-of-cycle indication onto the bus as it drives SCO\* active on the bus. On sensing SCO\*, all non-bus-owner agents enter a TRANSFER-CYCLE state.

# PARALLEL SYSTEM BUS SPECIFICATION

The TRANSFER-CYCLE state inhibits all non-bus-owner agents from arbitrating for access to the bus. All non-bus-owner agents remain in the TRANSFER-CYCLE state until the bus owner transmits an end-of-cycle indication onto the bus. The end-of-cycle indication occurs as SC2\* LOW (indicating that the transfer cycle is done) and SC4\* LOW (indicating that the replying agent is ready). As a result of the end-of-cycle indication, all agents enter the NO-TRANSFER-CYCLE state where they may assert a bus request and enter into arbitration for bus access.

All active agents on the bus must continually monitor the bus and stay in sync with the TRANSFER-CYCLE/NO-TRANSFER-CYCLE state-flow.



x-540

(EXCEPTION) = TIMEOUT\*=L OR BUSERR\*=L

(NO CYCLE) = ((NO-TRANSFER-CYCLE state) AND SC0\*=H) OR  
 ((TRANSFER-CYCLE state) AND SC2\*=L AND SC4\*=L) OR  
 (EXCEPTION)

(RESET) = RST\* or RSTNC\*

Figure 3-34. State-Flow Diagram For Requesting Agents Monitoring The Bus

## PARALLEL SYSTEM BUS SPECIFICATION

### 3.6.3 State-Flow Sequence For An Arbitration Cycle

Figure 3-35 presents a state-flow diagram for requesting agents performing an arbitration cycle. Replying agents do not perform arbitration cycles since they cannot become bus owners.

Figure 3-35 includes five operating states for a requesting agent in arbitration, as follows: the NO-REQUEST state, the RESOLUTION-1 state, the RESOLUTION-2 state, the RESOLUTION-3 state, and the BUS-OWNER state. The three states of the resolution may be thought of as one state; the transitions among them are not the result of signal transitions on the bus. The three resolution states are shown only to clarify the resolution phase. The following text describes how a requesting agent steps through each of the states shown in Figure 3-35.

All agents begin an arbitration cycle in the NO-REQUEST state. In that state, each agent is monitoring the state of the bus as described previously in Figure 3-34. Each agent requires no bus resources and asserts no requests.

The agent moves from the NO-REQUEST state to the RESOLUTION-1 state in Figure 3-35 by one of two methods:

- 1) the agent asserts a local signal, LBREQ, to request access to the Parallel System bus and the BREQ\* signal on the bus is not active. Agents must remove the LBREQ signal when they become the bus owner. The equation is:

$$\text{LBREQ} = \text{HIGH} \quad \text{AND} \quad \text{BREQ}^* = \text{HIGH}$$

- 2) the agent asserts a local signal, HPRI, to place a high priority request for bus access and the bus is not currently controlled by another agent. The equation is:

$$\text{HPRI} = \text{HIGH} \quad \text{AND} \quad (\text{EOC HANDSHAKE}) = \text{HIGH}$$

where EOC Handshake occurs when the current bus owner is about to exchange the bus.

From the RESOLUTION-3 state, the agent continues to drive its ARB and BREQ lines. When the bus acquisition conditions are met, the agent finds one of two conditions true:

- 1) WIN = H     The agent wins arbitration for the next bus access.  
The agent immediately becomes the bus owner.
- 2) WIN = L     The agent does not win the arbitration sequence. The agent must restart arbitration to try and gain access.

While in the RESOLUTION state, an agent drives the BREQ\* signal and the arbitration lines.

## PARALLEL SYSTEM BUS SPECIFICATION

A bus owner exits the BUS-OWNER state only on completing an operation and sensing a request for bus access (from another agent). At that time, the bus owner releases the bus and enters the NO-REQUEST state. If no other agent requests access to the bus, the bus owner retains ownership of the bus and performs its next operation.

At any given instant there may be one agent in the BUS-OWNER state. All other agents successful in entering the arbitration cycle are in the RESOLUTION 1, 2, or 3 states.

### 3.6.4 State-Flow Sequence For Requesting Agents In A Transfer Cycle

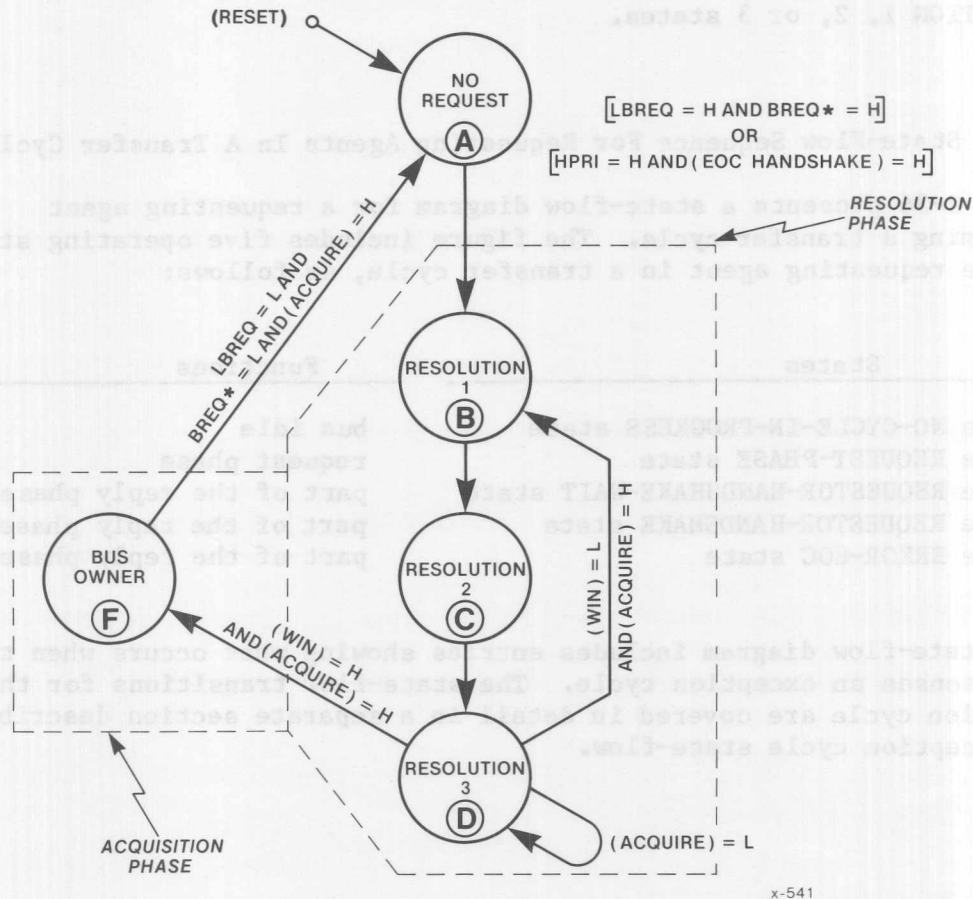
Figure 3-36 presents a state-flow diagram for a requesting agent performing a transfer cycle. The figure includes five operating states for the requesting agent in a transfer cycle, as follows:

States	Functions
the NO-CYCLE-IN-PROGRESS state	bus idle
the REQUEST-PHASE state	request phase
the REQUESTOR-HANDSHAKE-WAIT state	part of the reply phase
the REQUESTOR-HANDSHAKE state	part of the reply phase
the ERROR-EOC state	part of the reply phase

This state-flow diagram includes entries showing what occurs when the agent senses an exception cycle. The state-flow transitions for the exception cycle are covered in detail in a separate section describing the exception cycle state-flow.



# PARALLEL SYSTEM BUS SPECIFICATION



x-541

(EOC Handshake) = ((TRANSFER-CYCLE state from Figure 3-34) AND  
SC1\*=H AND SC2\*=L AND SC4\*=L)

(ACQUIRE) = (NO-CYCLE output from Figure 3-34) AND SC1\*=H

(WIN) = ARB5\*-ARB0\* signals match internal ID in the  
arbitration circuitry

(RESET) = RST\*=L OR RSTNC\*=L OR TIMOUT\*=L OR BUSERR\*=L

Figure 3-35. State-Flow Diagram For A Requesting Agent  
In An Arbitration Cycle

## PARALLEL SYSTEM BUS SPECIFICATION

The following text describes how a requesting agent steps through each of the states of the transfer cycle as shown in Figure 3-36.

Initially, a requesting agent begins a transfer cycle in the NO-CYCLE-IN-PROGRESS state. In this state, the agent is attempting to become the bus owner (part of the arbitration cycle state-flow). An agent exists in this state only when not performing a transfer cycle or an exception cycle; arbitration cycles occur simultaneously and are controlled via a separate state-flow (see Figure 3-34).

An agent has one path from the NO-CYCLE-IN-PROGRESS state; it must become the bus owner (as defined in the state-flow for the arbitration cycle) to begin the transfer cycle. On becoming the bus owner, the agent enters the REQUEST-PHASE state of the state-flow diagram for the transfer cycle. In that state, the requesting agent performs the request phase portion of a transfer cycle: the agent drives command information (via the SC9\* through SC0\* lines) and address information (via AD31\* through AD0\*) onto the bus.

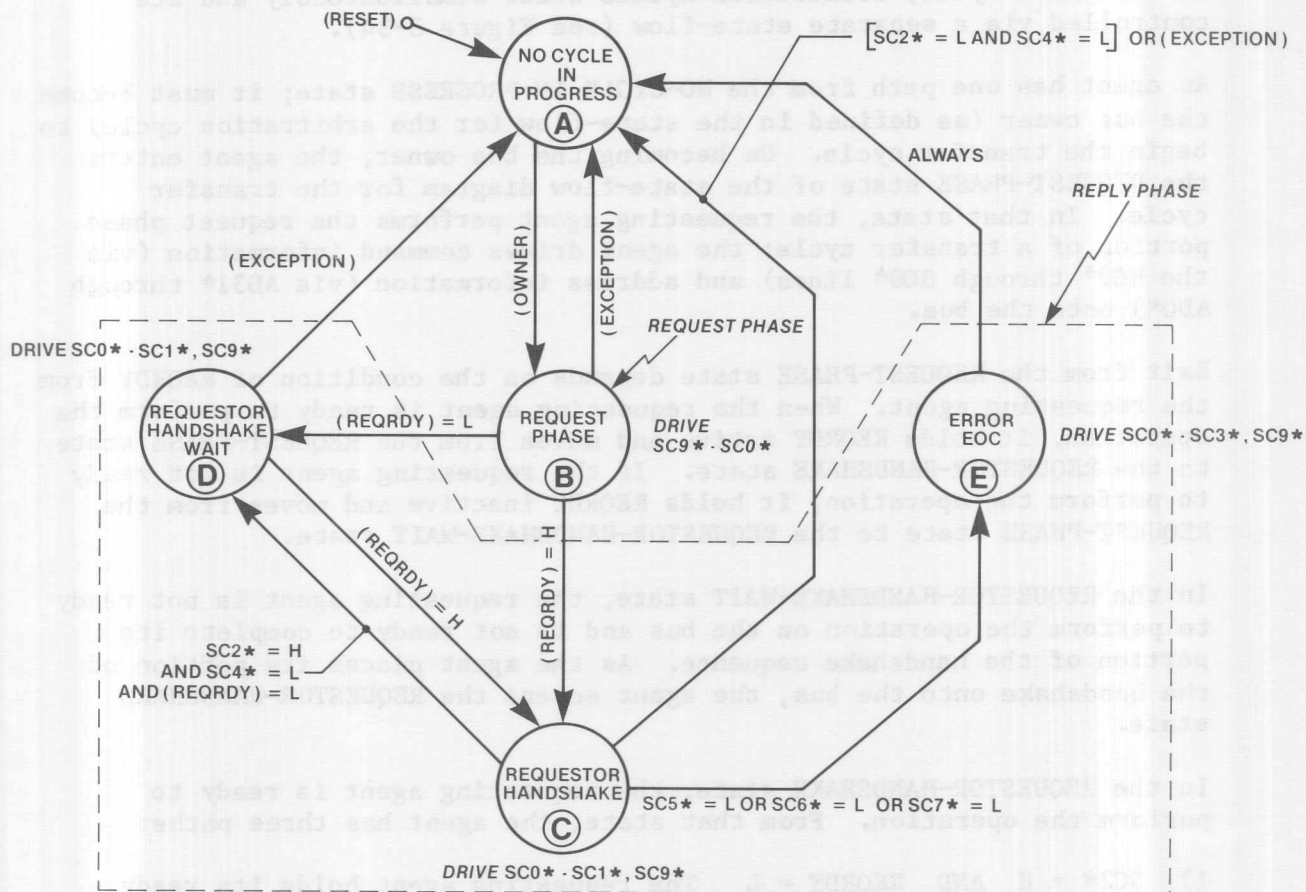
Exit from the REQUEST-PHASE state depends on the condition of REQRDY from the requesting agent. When the requesting agent is ready to perform the operation, it holds REQRDY active and moves from the REQUEST-PHASE state to the REQUESTOR-HANDSHAKE state. If the requesting agent is not ready to perform the operation, it holds REQRDY inactive and moves from the REQUEST-PHASE state to the REQUESTOR-HANDSHAKE-WAIT state.

In the REQUESTOR-HANDSHAKE-WAIT state, the requesting agent is not ready to perform the operation on the bus and is not ready to complete its portion of the handshake sequence. As the agent places its portion of the handshake onto the bus, the agent enters the REQUESTOR-HANDSHAKE state.

In the REQUESTOR-HANDSHAKE state, the requesting agent is ready to perform the operation. From that state, the agent has three paths:

- 1) SC2\* = H AND REQRDY = L    The requesting agent holds its ready signal active, has more data to transfer sequentially, and does not receive a ready signal from the replying agent. In this case, the agent returns to the HANDSHAKE-WAIT state.  
    AND SC4\* = L
- 2) SC2\* = L AND SC4\* = L    The requesting agent holds its ready signal active, senses an EOC, and receives a ready signal from the replying agent. In this case, the agent completes the transaction without errors and enters the NO-CYCLE-IN-PROGRESS state.
- 3) Agent error signal active    The requesting agent holds its ready signal active, but senses an agent error indication on the bus. The agent activates SC5\*, SC6\*, and SC7\* to identify the agent error and enter the EOC-EXCEPTION state and then immediately into the NO-CYCLE-IN-PROGRESS state.

# PARALLEL SYSTEM BUS SPECIFICATION



x-542

(OWNER) = ((RESOLUTION-3 state) AND (ACQUIRE)=H AND (WIN)=H) OR ((NEXT-BUS-OWNER state) AND (ACQUIRE)=H)

(EXCEPTION) = BUSERR\*=L OR TIMEOUT\*=L

(REQRDY) = Internal data-valid timing for read and write on bus

(RESET) = RST\*=L OR RSTNC\*=L

Figure 3-36. State-Flow Diagram For Requesting Agents In A Transfer Cycle

## PARALLEL SYSTEM BUS SPECIFICATION

The agent enters the ERROR-EOC state of the diagram by sensing either an agent error signal or a data width error signal from the replying agent. Either error causes the agent to leave the REQUESTOR-HANDSHAKE state and enter the ERROR-EOC state. In that state, an agent immediately gets off the bus by sending EOC and requestor ready signals (both SC2\* and SC3\* active). Following that, an agent enters the NO-CYCLE-IN-PROGRESS state and waits for the next bus operation.

### 3.6.5 State-Flow Diagram For Replying Agents In A Transfer Cycle

Figure 3-37 presents a state-flow diagram for replying agents responding to a transfer cycle. The figure includes four operating states for a replying agent:

States	Functions
the WAIT-FOR-REQUEST state	bus idle
the ADDRESS-DECODE state	the request phase
the REPLIER-HANDSHAKE state	part of the reply phase
the REPLIER-HANDSHAKE-WAIT state	part of the reply phase

This state-flow diagram includes entries showing what occurs when the agent senses an exception cycle. The state-flow transitions for the exception cycle are covered in detail in a separate section describing the exception cycle state-flow. The following text describes how a replying agent steps through each of the states shown in Figure 3-37.

The replying agent waits in the WAIT-FOR-REQUEST state until it receives the signal on the bus that indicates the beginning of a request phase of a transfer cycle (SC0\* = L). On sensing the request phase, all replying agents on the bus enter into the ADDRESS-DECODE state.

In the ADDRESS-DECODE state, each replying agent has three options depending on whether the address on the bus matches the address of the agent (ADDR = H when they match) and whether the agent is ready to transfer data (REPRDY = L when ready). The three options are:

- 1) ADDR = L  
The address on the bus does not match the address of the agent. The agent returns immediately to the WAIT-FOR-REQUEST state to wait for the next request phase.
- 2) ADDR = H AND REPRDY = L  
The address on the bus matches that of the agent and the replying agent is not ready to begin the data transfer. The replying agent begins the reply phase of a transfer cycle. The agent enters the REPLIER-HANDSHAKE-WAIT state.



## PARALLEL SYSTEM BUS SPECIFICATION

- 3) ADDR = H AND REPRDY = H The address on the bus matches that of the agent and the replying agent is ready to begin the data transfer. The replying agent begins the reply phase of a transfer cycle. The agent enters the REPLIER-HANDSHAKE state and begins driving SC7\* through SC4\*.

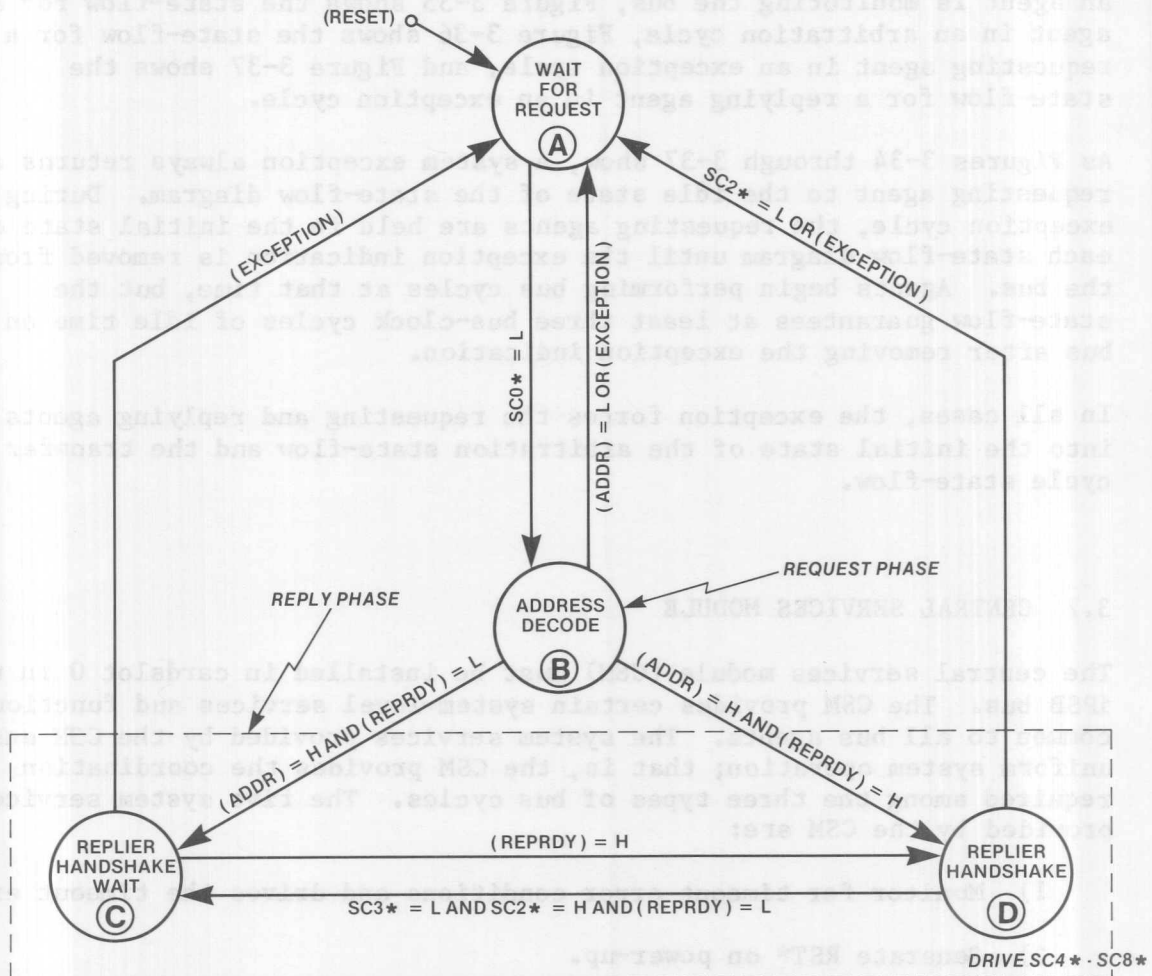
The replying agent enters the REPLIER-HANDSHAKE-WAIT when the replying agent is not ready. The inactive REPRDY signal places the replying agent into the REPLIER-HANDSHAKE-WAIT state from either the ADDRESS-DECODE state or the REPLIER-READY state. When the replying agent is ready for the data transfer, it activates the SC4\* line to complete its portion of the handshake. In doing so, the replying agent enters the REPLIER-HANDSHAKE state.

The replying agent waits in the REPLIER-HANDSHAKE state until the requesting agent completes its portion of the handshake. When both portions of the handshake are valid at the same time, the agents perform the data transfer.

If the replying agent senses no errors in the transfer, the agent waits in the REPLIER-HANDSHAKE state for the EOC indication on SC2\*. On receiving an EOC indication on SC2\*, the replying agent returns to the WAIT-FOR-REQUEST state to wait for the request phase of the next transfer cycle. If the replying agent receives no EOC indication, the replying agent remains in the REPLIER-HANDSHAKE state, assumes that a sequential transfer operation is being performed, and waits for additional data.

If the replying agent detects an agent error, the agent returns an error signal while in the REPLIER-HANDSHAKE state of the state-flow. The replying agent places an error status (SC5\*, SC6\*, and SC7\*) onto the Parallel System bus and waits in the REPLIER-HANDSHAKE state until the requesting agent provides an EOC signal. On receiving the EOC signal, the replying agent deactivates the error indication and returns the WAIT-FOR-REQUEST state.

# PARALLEL SYSTEM BUS SPECIFICATION



(ADDR) = address on bus during request phase matches address of agent

(EXCEPTION) = BUSERR\*=L OR TIMOUT\*=L

(REPRDY) = Internal data-valid timing for read and write on bus

(RESET) = RST\*=L OR RSTNC\*=L

Figure 3-37. State-Flow Diagram For Replying Agents In A Transfer Cycle

## 3.6.6 State-Flow Diagrams For Exception Cycles

An agent can detect an exception condition during a transfer cycle and, as a result, terminate both the transfer and arbitration cycles and begin an exception cycle. The state-flow diagram for an agent beginning an exception cycle is similar to that of the transfer cycle. Figure 3-34 shows the state-flow transitions caused by an exception that occurs while an agent is monitoring the bus, Figure 3-35 shows the state-flow for an agent in an arbitration cycle, Figure 3-36 shows the state-flow for a requesting agent in an exception cycle, and Figure 3-37 shows the state-flow for a replying agent in an exception cycle.

As Figures 3-34 through 3-37 show, a system exception always returns a requesting agent to the idle state of the state-flow diagram. During the exception cycle, the requesting agents are held in the initial state of each state-flow diagram until the exception indication is removed from the bus. Agents begin performing bus cycles at that time, but the state-flow guarantees at least three bus-clock cycles of idle time on the bus after removing the exception indication.

In all cases, the exception forces the requesting and replying agents into the initial state of the arbitration state-flow and the transfer cycle state-flow.

## 3.7 CENTRAL SERVICES MODULE

The central services module (CSM) must be installed in cardslot 0 in the iPSB bus. The CSM provides certain system-level services and functions common to all bus agents. The system services provided by the CSM ensure uniform system operation; that is, the CSM provides the coordination required among the three types of bus cycles. The five system services provided by the CSM are:

- 1) Monitor for timeout error conditions and drives the timeout error.
- 2) Generate RST\* on power-up.
- 3) Assign an arbitration ID to each agent on power-up of the system. The CSM assigns a separate and unique arbitration ID to each agent in the iPSB bus. This initialization occurs during the 50 ms reset operation. When DCLOW\* and PROT\* are inactive while RST\* is active. This allows agents time to prepare for the operation.
- 4) Assign a cardslot ID to each cardslot in the system. The CSM assigns a separate and unique cardslot ID to each agent in the iPSB bus. This initialization occurs during the 50 ms reset operation. When DCLOW\* and PROT\* are inactive while RST\* is active. However, at least one BCLK\* must elapse before the CSM sends the first cardslot ID onto the bus. This allows agents time to prepare for the operation.
- 5) Provides a central source for BCLK\* and CCLK\*.

# PARALLEL SYSTEM BUS SPECIFICATION

## 4. ELECTRICAL CHARACTERISTICS

### 4.1 GENERAL

This section provides the electrical characteristics and connector pin assignments for the signals on the Parallel System bus.

### 4.2 AC TIMING SPECIFICATIONS

All agents with interfaces to the Parallel System bus must adhere to two general categories of timing requirements: requirements for boards driving signals and requirements for boards receiving signals. In both cases the timing requirements are given with respect to the system-wide bus clock (BCLK\*) received at the specific board. The timing diagrams previous presented in this section have been amended to show relevant timing parameters. Figures 4-1 through 4-6 and Tables 4-1 through 4-9 provide timing specifications for the Parallel System bus.

All timing parameters listed or shown in this section are in nanoseconds unless otherwise specified. The maximum bus trace length between any two agents must be less than or equal to 16 inches. The maximum trace length on the bus is 16.8 inches. The maximum stub length on agents interfacing to the bus must be less than 2.5 inches.

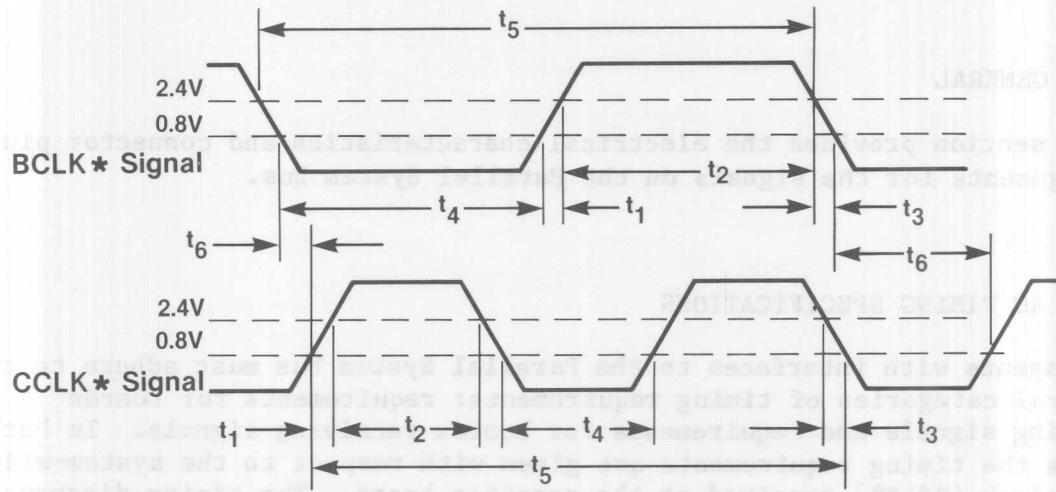
Table 4-1. Clock Specification

Parameter Number	Parameter Description	BCLK*		CCLK*	
		Min.	Max.	Min.	Max.
t1 (see Figure 4-1)	Rise Time	2	15	2	15
t2 (see Figure 4-1)	High Time	30	infinite	10	infinite
t3 (see Figure 4-1)	Fall Time	2	10	2	10
t4 (see Figure 4-1)	Low Time	30	infinite	10	infinite
t5 (see Figure 4-1)	Period	99.9	DC	49.95	DC
t6 (see Figure 4-1)	Clock-To-Clock	-10	+10	(see note 2)	

- Notes: 1. All measurements in nanoseconds unless otherwise noted.  
2. The frequency of CCLK\* is twice that of BCLK\*.

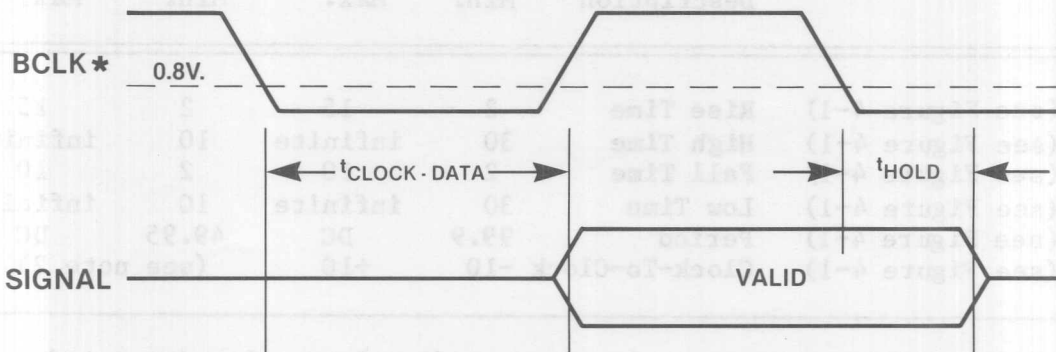


# PARALLEL SYSTEM BUS SPECIFICATION



x-554

Figure 4-1. BCLK\* And CCLK\* Timing Relationship



x-552

Figure 4-2. Driver Timing Parameters

# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-2. Timing Parameters For Signal Driver

Signal Names	$t_{\text{Clock-to-Data}}$ Maximum	$t_{\text{Hold}}$ Minimum	$t_{\text{off}}$ Maximum
AD31* - ADO*	40	10	40
PAR3* - PAR0*	40	10	40
SC9* - SCO*	40	8	30
BREQ*	40	8	40
ARB5* - ARB0*	45	8	45
TIMOUT*	40	8	40
BUSERR*	40	8	40
RST*	40	8	40
RSTNC*	40	8	40

- Notes: 1. The minimum  $t_{\text{CLOCK-DATA}}$  must be greater than or equal to the minimum  $t_{\text{HOLD}}$ .
2. The maximum slew rate on all signals on the bus must be less than or equal to 0.5 volts per nanosecond.
3.  $T_{\text{off}}$  is the turn-off time.
4. All of the parameters are specified at the driver for a 50pF capacitive load over the temperature and voltage range of the driver.

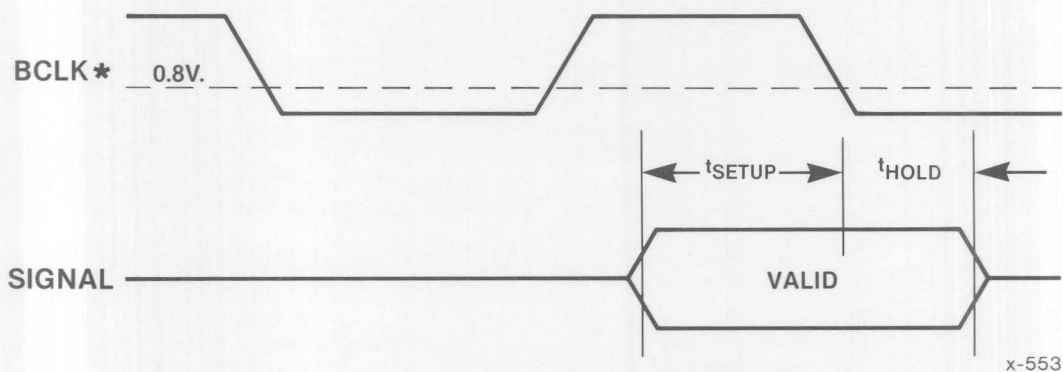


Figure 4-3. Receiver Timing Parameters

# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-3. Timing Parameters For Signal Receivers

Signal Names	$t_{\text{SETUP}}$ Minimum	$t_{\text{HOLD}}$ Minimum
AD31* - ADO*	30	5
PAR3* - PAR0*	30	5
SC9* - SC0*	30	3
BREQ*	21	3
ARB5* - ARB0*	10	3
TIMOUT*	30	3
BUSERR*	21	3
RST*	30	3
RSTNC*	21	3

- Notes: 1. The amount of time lost due to backplane transmission line requirements has been taken into account in calculating the  $t_{\text{SU}}$  times from the  $t_{\text{CD}}$  times. The time delay is the sum of two bus propagation delays (25 ns) plus the maximum clock skew (5 ns) for all signals except the open-collector and SC9\*-SC0\* lines. These add to a total bus loss of 30 ns for non-open-collector signals and for the SC9\*-SC0\* signals. The bus loss for open-collector signals is 39 ns. The SC9\*-SC0\* signals have a bus loss of 30 ns when driven and have a bus loss of 39 ns when not driven.
2. The maximum clock skew between any two agents must be less than or equal to 5 nanoseconds.



# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-4. Cold-Start Control Timing

Parameter	Description	Min.	Max.	Units
t1 (see Figure 4-4)	DC power set-up to DCLOW*	---	1	msec
t2 (see Figure 4-4)	Cold-reset duration	2.5	---	msec
t3 (see Figure 4-4)	Warm-reset duration	50	---	msec
t4 (see Figure 4-4)	RSTNC* set-up to RST*	1	---	BCLK period

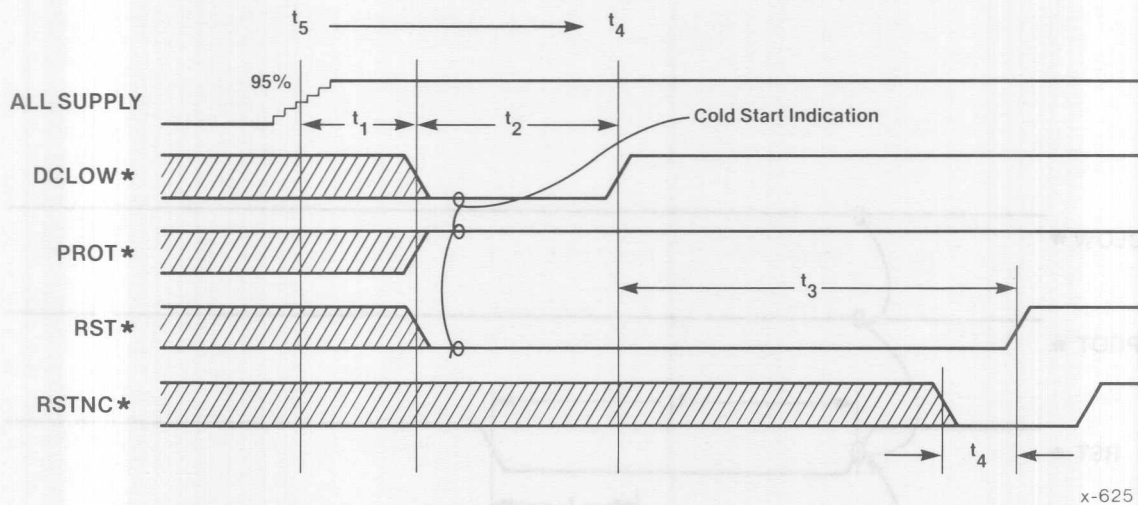


Figure 4-4. Cold-Start Timing Parameters



# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-5. Warm-Start Control Timing

Parameter	Description	Min.	Max.	Units
t1 (see Figure 4-5)	RST* pulse width (CSM)	50	---	msec
t2 (see Figure 4-5)	RSTNC* set-up to RST*	1	---	BCLK period

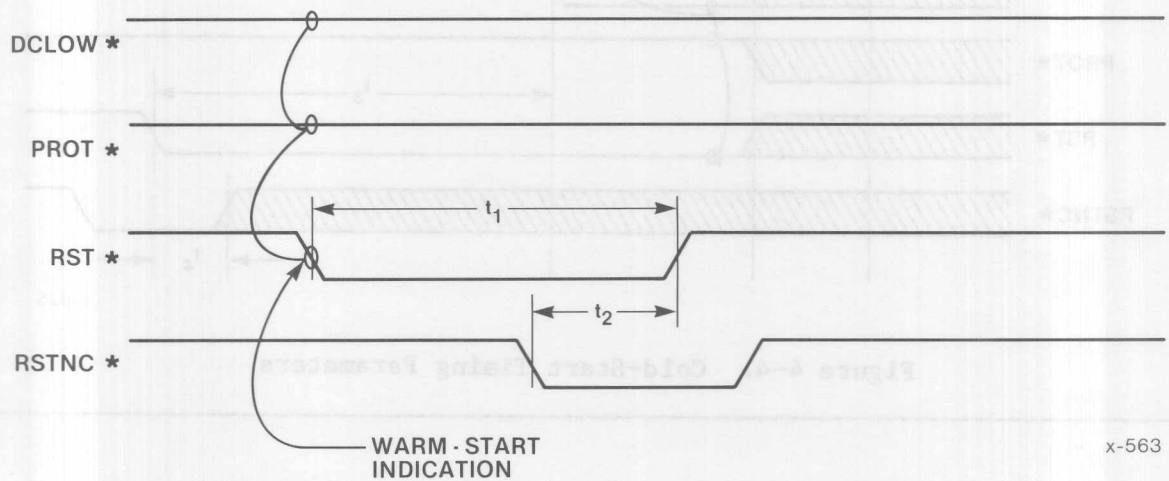


Figure 4-5. Warm-Start Timing Parameters

# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-6. Power Failure And Recovery Timing

Parameter	Description	Min.	Max.	Units
t1 (see Figure 4-6)	DC power hold from DCLOW*	6.5	---	msec
t2 (see Figure 4-6)	PROT* delay from DCLOW*	6.0	6.25	msec
t3 (see Figure 4-6)	DC power set-up to DCLOW*	1	---	msec
t4 (see Figure 4-6)	RST* delay from DCLOW*	6.5	7.0	msec
t5 (see Figure 4-6)	RST* set-up from DCLOW*	0.5	---	msec
t6 (see Figure 4-6)	RST* active from PROT*	50	---	msec
t7 (see Figure 4-6)	DCLOW* pulse width	7.5	---	msec
t8 (see Figure 4-6)	PROT* hold from DCLOW*	2.0	2.5	msec
t9 (see Figure 4-6)	RSTNC* set-up to RST*	1	---	BCLK period

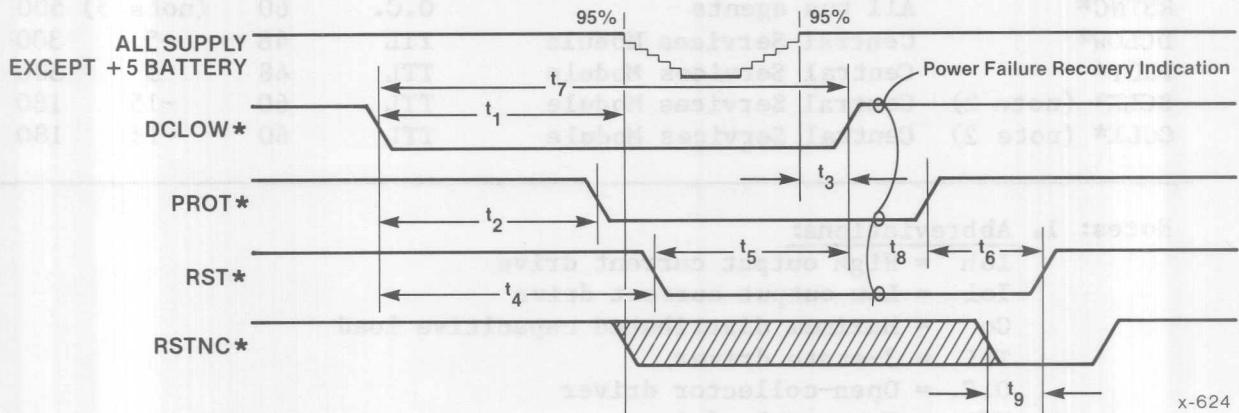


Figure 4-6. Power Failure And Recovery Timing Parameters

# PARALLEL SYSTEM BUS SPECIFICATION

## 4.3 DC SPECIFICATIONS FOR SIGNALS

All agents on the Parallel System bus must adhere to the DC requirements presented in Tables 4-7 through 4-9.

Table 4-7. DC Specifications For Signal Drivers

Signal Names	Signal Drivers/Locations	Signal Type	Iol (ma)	Ioh (ma)	Co (pf)
AD31*-AD0*	Requesting and replying agents	Tri	48	-5	500
PAR3*-PAR0*	Requesting and replying agents	Tri	48	-5	500
SC9*-SC0*	Requesting and replying agents	Tri	64	-15	500
BREQ*	Requesting agent	O.C.	60	(note 5)	500
ARB5*-ARBO*	All agents	O.C.	60	(note 5)	500
BUSERR*	Requesting and replying agents	O.C.	60	(note 5)	500
TIMOUT*	Central Services Module	TTL	48	-5	300
LACHn*	Connected to ADxx* lines and received only	(refer to the AD lines)			
RST*	Central Services Module	TTL	48	-5	300
RSTNC*	All bus agents	O.C.	60	(note 5)	500
DCLOW*	Central Services Module	TTL	48	-5	300
PROT*	Central Services Module	TTL	48	-5	300
BCLK* (note 2)	Central Services Module	TTL	60	-15	180
CCLK* (note 2)	Central Services Module	TTL	60	-15	180

Notes: 1. Abbreviations:

- Ioh = High output current drive
- Iol = Low output current drive
- Co = Maximum distributed capacitive load
- Tri = 3-state driver
- O.C. = Open-collector driver
- TTL = Totem-pole driver

2. The BCLK\* and CCLK\* signals must be driven from the middle of the backplane by two sets of drivers, each driving half of the backplane if the number of agents in the backplane exceeds twelve. The maximum skew between the two halves of either BCLK\* or CCLK\* must not exceed 1 nanosecond.
3. All three-state signals must be actively driven high when required.
4. The Iol and Ioh values are as follows:
  - Maximum Iol at 0.55 volts
  - Maximum Ioh at 2.40 volts
5. The high level output leakage (Ioh) of the open-collector signals must be less than or equal to 400 microamps at 5.25 volts.

# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-8. DC Specifications For Signal Receivers

Signal Names	Signal Receiver Locations	Iil+Iozl (ma)	Iih+Iozh (ua)	Ci (pf)
AD31*-AD0*	Requesting and replying agents	-1	100	20
PAR3*-PAR0*	Requesting and replying agents	-1	100	20
SC9*-SC0*	Requesting and replying agents	-1	100	20
BREQ*	Requesting Agents	-0.9	100	20
ARB5*-ARB0*	All agents	-0.9	100	20
BUSERR*	Requesting and replying agents	-0.9	100	20
TIMOUT*	Requesting agents	-1	100	20
LACHn*	On ADxx* lines for agents	-0.1	100	10
RST*	All bus agents	-1	100	12
RSTNC*	All bus agents	-0.9	100	20
DCLOW*	All bus agents	-1	100	12
PROT*	All bus agents	-1	100	12
BCLK* (note 3)	All bus agents	-1.5	100	12
CCLK* (note 3)	All bus agents	-1.5	100	12

Notes: 1. Abbreviations:

- Iih = High input current load
- Iil = Low input current load
- Iozl = Leakage for three-state high impedance low output.
- Iozh = Leakage for three-state high impedance high output.
- Ci = Capacitive load presented by driver/receivers.

2. The Iol and Ioh values are as follows:

- Maximum Iol at 0.55 volts
- Maximum Ioh at 2.40 volts

3. The BCLK\* and CCLK\* signals must be driven from the middle of the backplane by two sets of drivers, each driving half of the backplane if the number of agents in the backplane exceeds twelve. The maximum skew between the two halves of either BCLK\* or CCLK\* must not exceed 1 nanosecond.



# PARALLEL SYSTEM BUS SPECIFICATION

Table 4-9. Backplane Termination Requirements

Signal Names (bits)	Location Of Termination (ns)	Termination To 5V (ohms) (+ 5%)	Termination To Gnd (ohms) (+ 5%)
AD31*-AD0*	Both ends of backplane	330	470
PAR3*-PAR0*	Both ends of backplane	330	470
SC9*-SC0*	Both ends of backplane	220	330
BREQ*	Both ends of backplane	220	330
ARB5*-ARBO*	Both ends of backplane	220	330
BUSERR*	Both ends of backplane	220	330
TIMOUT*	Both ends of backplane	330	470
LACHn*	Not required (see ADxx* lines)	none	none
RST*	Both ends of backplane	330	470
RSTNC*	Both ends of backplane	220	330
DCLOW*	Both ends of backplane	330	470
PROT*	Both ends of backplane	330	470
BCLK*	Farthest point from driver	110	120
CCLK*	Farthest point from driver	110	120

# PARALLEL SYSTEM BUS SPECIFICATION

## 4.4 CURRENT LIMITATIONS PER CONNECTOR

Tables 4-10 and 4-11 give the voltage and current requirements for a system using only the Parallel System bus and for a system using both the Parallel System bus and the iLBX bus portions of the Multibus II architecture. The voltage specifications at the connector are measured over the full current range.

Table 4-10. Power Limitations For A One-Connector Agent  
(For An Agent Using Only The Parallel System Bus)

Minimum Volts	Nominal Volts	Maximum Volts	Maximum Amps
+4.75	+5.00	+5.25	9.0 (note 1)
+4.75	+5.00	+5.25	2.0 (Battery)
+11.40	+12.00	+12.60	2.0
-11.40	-12.00	-12.60	2.0
---	0 V	---	15.0

Note: 1. On all cardslots except the CSM (cardslot 0).

Table 4-11. Power Limitations For A Two-Connector Agent  
(For An Agent Using Both the iPSB™ Bus And The iLBX™ II Bus Connectors)

Minimum Volts	Nominal Volts	Maximum Volts	Maximum Amps
+4.75	+5.00	+5.25	15.0 (note 1)
+4.75	+5.00	+5.25	2.0 (Battery)
+11.40	+12.00	+12.60	2.0
-11.40	-12.00	-12.60	2.0
---	0 V	---	23.0

Note: 1. On all cardslots except the CSM (cardslot 0).

# PARALLEL SYSTEM BUS SPECIFICATION

## 4.5 PIN ASSIGNMENTS

The pin assignment for the 96-pin connector to the Parallel System bus is listed in Table 4-12.

Table 4-12. Parallel System Bus Connector Pinout

Connector Pin Number	Row A	Row B	Row C
1	0 Volts	PROT*	0 Volts
2	+5 Volts	DCLOW*	+5 Volts
3	+12 Volts	+5 Battery	+12 Volts
4	(note 1)Reserved	SDA(note 3)	BCLK*
5	TIMOUT*	SDB(note 3)	0 Volts
6	(note 2)LACHn*	Ground	CCLK*
7	AD00*	AD01*	0 Volts
8	AD02*	0 Volts	AD03*
9	AD04*	AD05*	AD06*
10	AD07*	+5 Volts	PAR0*
11	AD08*	AD09*	AD10*
12	AD11*	+5 Volts	AD12*
13	AD13*	AD14*	AD15*
14	PAR1*	0 Volts	AD16*
15	AD17*	AD18*	AD19*
16	AD20*	0 Volts	AD21*
17	AD22*	AD23*	PAR3*
18	AD24*	0 Volts	AD25*
19	AD26*	AD27*	AD28*
20	AD29*	0 Volts	AD30*
21	AD31*	Reserved	PAR3*
22	+5 Volts	+5 Volts	Reserved
23	BUSREQ*	RST*	BUSERR*
24	ARB5*	+5 Volts	ARB4*
25	ARB3*	RSTNC*	ARB2*
26	ARB1*	0 Volts	ARB0*
27	SC9*	SC8*	SC7*
28	SC6*	0 Volts	SC5*
29	SC4*	SC3*	SC2*
30	-12 Volts	+5 Battery	-12 Volts
31	+5 Volts	SC1*	+5 Volts
32	0 Volts	SC0*	0 Volts

### Notes:

- Slot 0: Second BCLK\* driver for systems containing more than 12 slots; drives BCLK\* (pin 4C) to left half of backplane.  
Slot 1-19: 0 volts.
- Slot 0: Second CCLK\* driver for systems containing more than 12 slots; drives CCLK\* (pin 6C) to left half of backplane.  
Slot 1-19: LACHn\*.
- Signal lines SDA and SDB are reserved for the Serial System Bus; refer to section 4.

## 5. COMPLIANCE LEVELS

## 5.1 INTRODUCTION

This section defines the variability allowed within agents on the Parallel System Bus (iPSB bus) portion of the Multibus II Bus Architecture. The purpose in defining the limits of variability is to assure the maximum amount of upward compatibility. In most cases, agents designed to different levels of compliance create a system with an over-all compliance of the least complex agent.

## 5.2 DATA PATH

The iPSB bus allows agents with 8-bit, 16-bit, and 32-bit data paths to co-exist on the data bus. The bus allows consecutive transactions that are directed to different agents of varied bus width. A system implementation that uses agents with more than one bus width for the data path must define the data width of devices properly within the interconnect space for each agent. Agents with higher levels of data transfer must support all lower levels of compliance.

## 5.3 ADDRESS PATH

The iPSB bus has a 32-bit address path. Agents on the bus must implement all 32-bits.

## 5.4 COMPLIANCE CODES

The codes assigned to the various areas of compliance for the iPSB bus are as follows:

o	Type of device - requesting agent	RQA
	- replying agent	RPA
	- both	RQA/RPA
o	Data Path Width - 16-bit	D16
	- 32-bit	D32
o	Message Support - 16 bit	M16
	- 32 bit	M32
	- none	-
o	CSM Module Support - on-board	C
	- not on-board	-
o	Sequential Transfer Support - on-board	S
	- none	-



# PARALLEL SYSTEM BUS SPECIFICATION

## 5.5 COMPLIANCE STATEMENT EXAMPLE

The compliance statement for an iPSB bus compatible board should be clearly defined in the board-level specification for the agent. Omission of any particular compliance code is interpreted as non-support of the capability.

As an example a requesting/replying agent that can perform 8-bit, 16-bit, and 32-bit data transfers and 16 or 32-bit message operations would be marked as follows:

RQA/RPA D8 D16 D32 M32

A replying agent that can perform only 32-bit data transfers with no message support would be marked as follows:

RPA D32

Compliance Code	Description	Agent Type
RQA	Requesting agent	Requesting agent
RPA	Replying agent	Replying agent
RQA/RPA	Both	Both
D8	8-bit data width	8-bit
D16	16-bit data width	16-bit
D32	32-bit data width	32-bit
M16	16-bit message support	16-bit
M32	32-bit message support	32-bit
-	None	None
C	On-board	On-board
-	Not on-board	Not on-board
S	Sequential transfer support	Sequential transfer support
-	None	None

\*\*\*



# INTEL iLBX™ II BUS SPECIFICATION

---







# CONTENTS

	PAGE
iLBX™ II BUS SPECIFICATION	
1.1 Scope.....	3-1
1.2 Object.....	3-1
1.3 Definitions.....	3-2
1.4 MULTIBUS® II Architecture Overview.....	3-2
1.5 iLBX™ II Bus Overview.....	3-3
1.5.1 Multiple Agent Support.....	3-5
1.5.2 Mutual Exclusion Capability.....	3-6
1.5.3 Pipelined Bus Operation.....	3-6
1.5.4 Pipelining on the iLBX™ II Bus.....	3-6
1.5.5 Two Address Spaces.....	3-7
1.5.6 Handshake Protocol.....	3-7
1.5.7 Bus Cycles on the iLBX™ II Bus.....	3-8
2. Signal Descriptions.....	3-10
2.1 General.....	3-10
2.2 Signal Naming and Notational Conventions.....	3-10
2.3 Signal Groups.....	3-11
2.3.1 Arbitration Cycle Signal Group.....	3-11
2.3.2 Transfer Cycle Signal Group.....	3-12
2.3.3 Exception Cycle Signal Group.....	3-15
2.3.4 System Control Signal Group.....	3-15
3. iLBX™ II Bus Protocol.....	3-19
3.1 General.....	3-19
3.2 Handshake Protocol.....	3-20
3.2.1 Access Delay Times of Replying Agents.....	3-20
3.2.2 Wait Signal Operation.....	3-22
3.3 iLBX™ II Bus Protocol.....	3-22
3.3.1 Arbitration Cycle.....	3-23
3.3.2 Transfer Cycle.....	3-25
3.3.3 Exception Cycle Protocol.....	3-37
3.4 Mutual Exclusion.....	3-39
3.5 State-Flow Diagrams.....	3-40
3.5.1 Arbitration Cycle.....	3-40
3.5.2 Transfer Cycle.....	3-41
4. Electrical Specifications.....	3-46
4.1 General.....	3-46
4.2 iLBX™ II Bus Signal Characteristics.....	3-46
4.3 iLBX™ II Bus DC and AC Loading Specifications.....	3-49
4.4 AC Timing Specifications.....	3-51
4.4.1 Signal Driver Requirements.....	3-51
4.4.2 Signal Receiver Requirements.....	3-54
4.4.3 Preferred Driver/Receiver Components.....	3-55



	PAGE
iLBX™ II BUS SPECIFICATION (continued)	

5. Compliance Levels.....	3-56
5.1 General.....	3-56
5.2 Data Path.....	3-56
5.3 Address Path.....	3-56
5.4 Compliance Codes.....	3-56
5.5 Compliance Statement Example.....	3-57

## TABLES

2-1. Signal Notation Summary.....	3-10
2-2. Signal Groups on the iLBX™ II Bus.....	3-11
2-3. iLBX™ II Bus Command Signal Encoding.....	3-13
3-1. Data Alignment Policy.....	3-36
3-2. Address Space Data Alignment Restrictions.....	3-37
4-1. iLBX™ II Bus Signal Type and Terminations.....	3-47
4-2. iLBX™ II Bus Signal Pin Assignments.....	3-48
4-3. iLBX™ II Bus AC and DC Loading Requirements.....	3-50
4-4. iLBX™ II Bus Driver Timing Specifications.....	3-52
4-5. iLBX™ II Bus Receiver Timing Specifications.....	3-55

## FIGURES

1-1. The MULTIBUS® II System Architecture.....	3-3
1-2. Maximum System Configuration on the iLBX™ II Bus.....	3-4
1-3. Block Diagram of the iLBX™ II Bus Interface.....	3-5
1-4. Pipelining on the iLBX™ II Bus.....	3-6
3-1. No Access Delay Replying Agent Single-Transfer Operation...	3-21
3-2. One-Clock Access Delay in a Replying Agent Performing a Single-Transfer Operation.....	3-21
3-3. Bus Cycles.....	3-22
3-4. Bus Ownership Exchange, Primary-to-Secondary.....	3-25
3-5. Transfer Cycle on the iLBX™ II Bus.....	3-26
3-6. Organization of Address Spaces and Addressing Formats.....	3-28
3-7. READ Operation Reply Phase.....	3-30
3-8. Block WRITE Operation Reply Phase.....	3-31
3-9. Block READ Operation Reply Phase.....	3-32
3-10. Data Alignment and Access Space-to-Bus Mapping.....	3-34
3-11. Data Alignment Interface Requirements.....	3-35
3-12. Exception Cycle.....	3-38
3-13. Locked Operation.....	3-39



## CONTENTS (continued)

### FIGURES (continued)

	PAGE
iLBX™ II BUS SPECIFICATION (continued)	
3-14. State-Flow Diagram for the Arbitration in the Primary Requesting Agent.....	3-41
3-15. State-Flow Diagram for Requesting Agents.....	3-43
3-16. Bus Interface State-Flow for a Replying Agent.....	3-45
4-1. Driver Timing Parameter Definition.....	3-51
4-2. Bus Clock Timing Parameters.....	3-53
4-3. iLBX™ II Receiver Timing Parameter Definitions.....	3-54



FIGURES (continued)

PAGE

APPENDIX II BUS SPECIFICATION (continued)

3-14	State-Flow Diagram for the Arbitration in the Primary Requesting Agent.....	3-14
3-15	State-Flow Diagram for Requesting Agents.....	3-15
3-16	Bus Interface State-Flow for a Receiving Agent.....	3-16
4-1	Driver Timing Parameter Definition.....	4-1
4-2	Bus Clock Timing Parameters.....	4-2
4-3	APPENDIX II Receiver Timing Parameter Definitions.....	4-3



## CHAPTER 3 iLBX™ II BUS SPECIFICATION

### 1.1 SCOPE

This specification defines a method of interfacing microprocessor computer components via the Local Bus Extension (iLBX II) interface portion of the Multibus II bus architecture.

This specification gives a description of the functions, attributes, and operation of the iLBX II bus portion of the Multibus II bus architecture. The Local Bus Extension interface is a high-speed parallel interface that provides an extension of on-board memory facilities. The iLBX II interface presents two advantages: it adds local memory that does not require more room on the original CPU board and it allows the local CPU, in accessing the additional memory, to achieve a performance level comparable to that of accessing local on-board memory.

The specification applies to microprocessor computer systems or portions of them where:

- 1) the data transmission rate does not exceed 48 megabytes per second, the maximum sustained bandwidth.
- 2) the length of the data path on the backplane does not exceed 5.0 inches.
- 3) the data exchanged among units is digital (rather than analog).
- 4) the total number of units connected to one contiguous bus does not exceed 6.

### 1.2 OBJECT

This standard is intended:

- 1) To define a high bandwidth, low latency, local memory expansion interface for use with microcomputer equipment.
- 2) To specify the electrical and functional interface requirements that each connected unit shall meet in order to be interconnected to and communicate on the iLBX II bus.
- 3) To specify terminology and definitions related to the iLBX II bus.
- 4) To guide independent manufacturers in designing computer equipment that is architecturally compatible with the iLBX II bus.



### 1.3 DEFINITIONS

Many of the definitions for the iLBX II bus also apply to the Parallel System Bus and are not repeated here. The following definitions apply only to the iLBX II Bus Specification. More specific definitions are provided in other sections of this specification, as appropriate.

**Primary  
Requesting  
Agent**

A required type of agent that normally has immediate access to the iLBX II bus when the bus is not controlled by the secondary requesting agent. The primary requesting agent must have an interface to the iPSB bus.

**Secondary  
Requesting  
Agent**

An optional, secondary agent that normally does not have immediate control of the iLBX II bus and requests bus access from the primary requesting agent. The secondary requesting agent may have an interface to the iPSB bus.

**Access Delay**

The minimum number of bus clocks required for a replying agent to return data for an operation.

**Block Transfer**

A bus operation in which a single request phase (one address/command) returns from two or more data during the reply phase.

**Data Transfer  
Period**

A clock cycle during which valid data is accepted by either the replying agent or the requesting agent.

### 1.4 MULTIBUS® II ARCHITECTURE OVERVIEW

Figure 1-1 shows the Multibus II bus architecture and shows how the iLBX II bus contributes to that architecture.

As Figure 1-1 shows, the Multibus II bus architecture provides three separate busses: the Parallel System Bus (iPSB bus), the Local Execution Bus (iLBX II bus), and the Serial System Bus (iSSB bus). In some cases the functions of busses overlap, however, each bus is optimized to add a particular facet to the Multibus II architecture. The three busses provide you with the ability to put together the system environment that best suits the cost/performance and bandwidth/latency goals for an application.

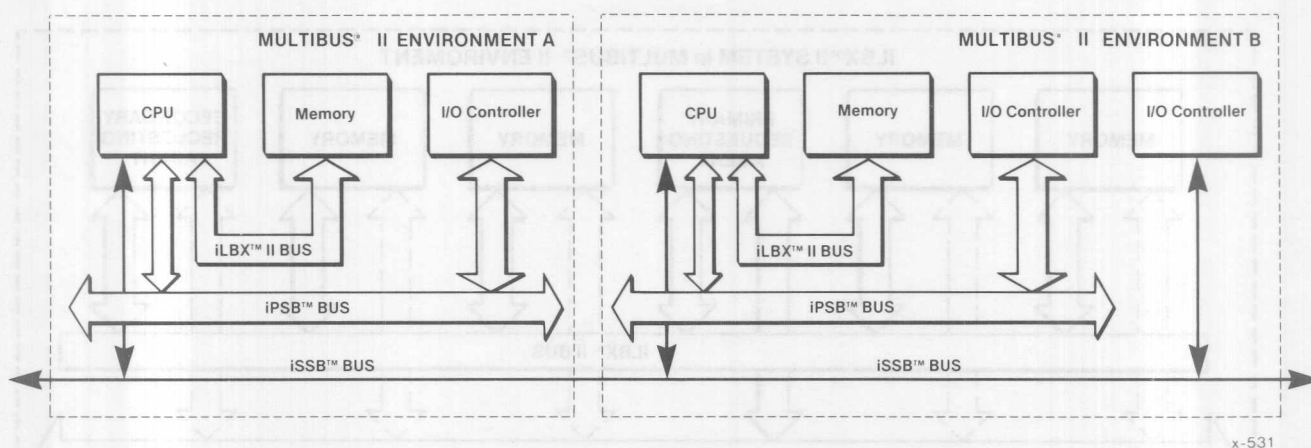


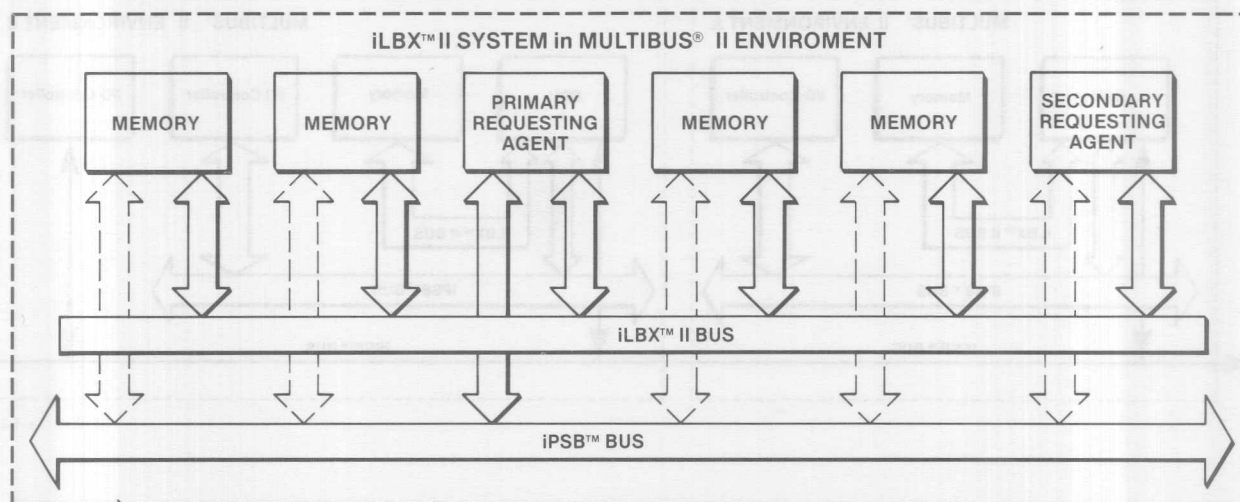
Figure 1-1. MULTIBUS® II System Architecture

### 1.5 iLBX™ II BUS OVERVIEW

The iLBX II bus is a high-speed execution bus that is separate from the iPSB bus and dedicated to use by only one or two CPU boards. Figure 1-2 shows a block diagram of the maximum iLBX II system.

The iLBX II system may consist of up to two requesting agents, but one of them must be a primary requesting agent. In addition, the system may contain up to four replying agents (five if the secondary requesting agent is not installed). As the figure shows, an interface to the iPSB or iSSB bus is required on the primary requesting agent and may be present on each of the other agents, but is not required.

One iLBX II bus interface can be extended to a maximum of 6 agents and 5.0 inches. A system requiring more than 6 agents may contain more than one iLBX II bus interface, however, the iLBX II bus interfaces must remain separate within the application.



x-608

Figure 1-2. Maximum System Configuration on the iLBX™ II Bus

Figure 1-3 shows a diagram of the iLBX II bus. As the figure shows, the iLBX II bus consists of four groups of signals that requesting agents and replying agents use to communicate with one another: the arbitration signals, the transfer cycle signals, the exception cycle signals, and the system control signals. The iLBX II bus requires no signals from the iPSB bus for operation.

The iLBX II bus has several specific attributes that make it a unique and independent part of the Multibus II system architecture, as follows:

- 1) The bus allows up to six agents, two of which may be requesting agents.
- 2) The bus provides a mutual exclusion capability to control multi-ported replying agents.
- 3) The separate address, command, and data paths on the bus provide for pipelined operation of the bus; the bus can contain both address/command information and data information.
- 4) The bus operates synchronous with the bus clock, XBCLK\*, at a maximum rate of 12 MHz.
- 5) The bus uses a uni-directional handshake protocol to achieve fast operation.

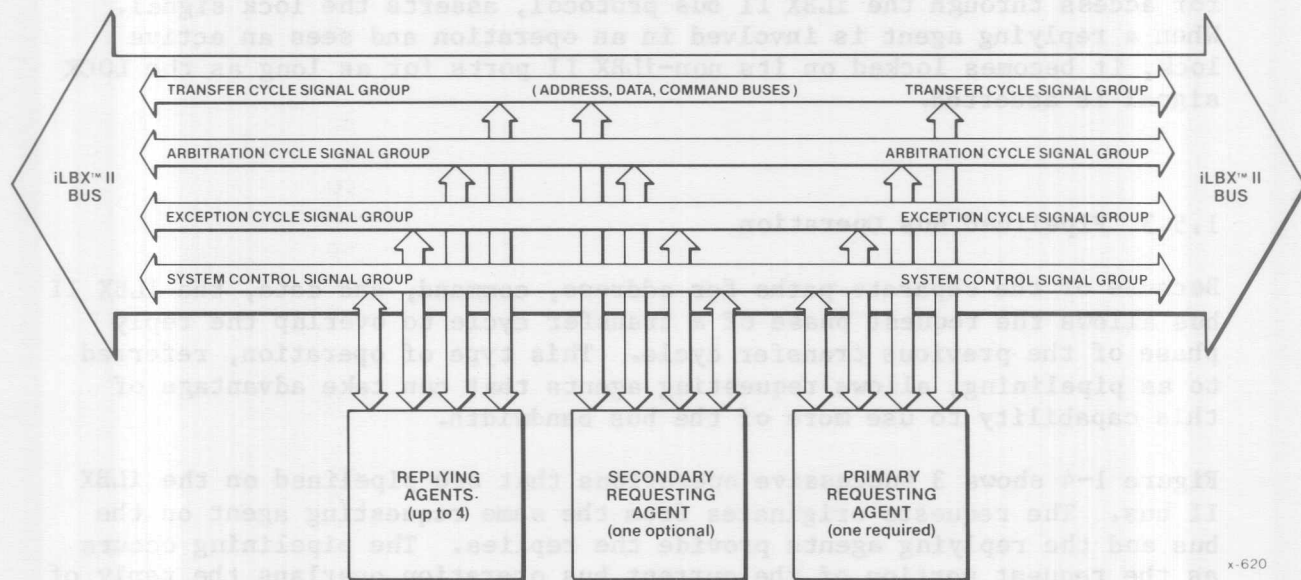


Figure 1-3. Block Diagram Of The iLBX™ II Bus Interface

- 6) The bus provides access to two address spaces: memory space and interconnect space. These space may be combined with or separated from the similar address spaces on the IPSB bus.
- 7) The bus operates in defined bus cycles. The three types of bus cycles are: arbitration cycle, transfer cycle, and exception cycle.

Each of these attributes is explained further in the following paragraphs.

#### 1.5.1 Multiple Agent Support

One iLBX II system is limited to six agents. A maximum of two of those agents can be requesting agents, referred to as the primary requesting agent and the secondary requesting agent. A primary requesting agent is required in all iLBX II systems; a secondary requesting agent is optional. The remaining agents in an iLBX II system are replying agents (i.e. memory boards). A typical Multibus II system could include more than one iLBX II system.



### 1.5.2 Mutual Exclusion Capability

The iLBX II bus includes a mutual exclusion mechanism that requesting agents can use in controlling access to a multi-ported replying agent. To access resources on a replying agent, a requesting agent arbitrates for access through the iLBX II bus protocol, asserts the lock signal. When a replying agent is involved in an operation and sees an active lock, it becomes locked on its non-iLBX II ports for as long as the LOCK signal is asserted.

### 1.5.3 Pipelined Bus Operation

Because of the separate paths for address, command, and data, the iLBX II bus allows the request phase of a transfer cycle to overlap the reply phase of the previous transfer cycle. This type of operation, referred to as pipelining, allows requesting agents that can take advantage of this capability to use more of the bus bandwidth.

Figure 1-4 shows 3 successive operations that are pipelined on the iLBX II bus. The requests originate from the same requesting agent on the bus and the replying agents provide the replies. The pipelining occurs as the request portion of the current bus operation overlaps the reply of the next operation.

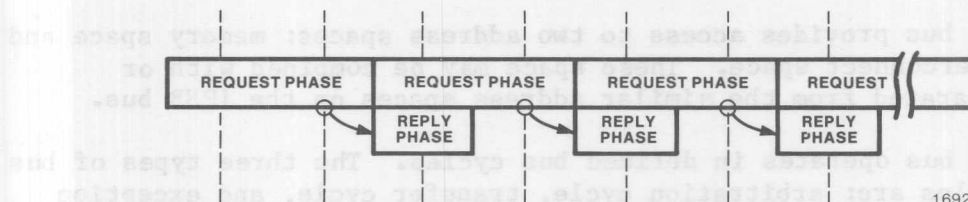


Figure 1-4. Pipelining On The iLBX™ II Bus

### 1.5.4 Synchronous Bus Operation

All events performed on the iLBX II bus are synchronous to a reference timing signal called the bus clock. All bus operations are defined in terms of the bus clock cycle; the start, the intermediate phases, and the end of each bus operation has a defined relationship to the edge of the bus clock. Figure 1-5 shows the relationship between the bus clock and the bus operations.

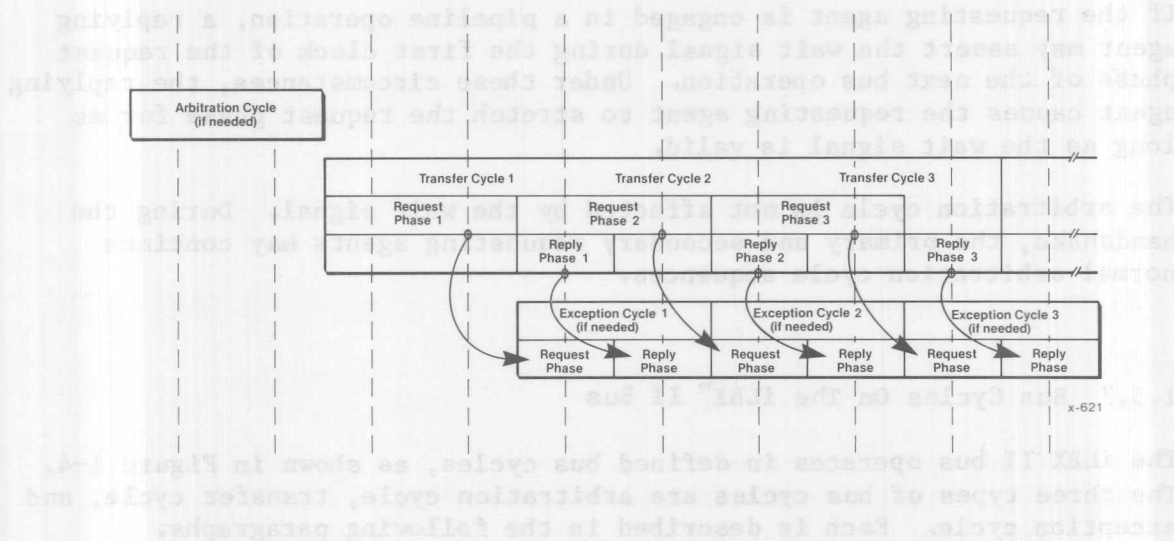


Figure 1-5. Relationship Among Cycles And Bus Clock

### 1.5.5 Two Address Spaces

The iLBX II bus provides two address spaces: memory space and interconnect space. Requesting agents use the memory space to access the memory resources on the iLBX II bus. The interconnect space allows the primary requesting agent on the iLBX II bus to initialize and configure all other agents on the bus. The interconnect space for the iLBX II bus may be separated from that of the iPSB bus, depending on the implementation of the Multibus II system.

### 1.5.6 Handshake Protocol

The protocol on the iLBX II bus employs a uni-directional handshake between requesting and replying agents. The replying agent(s) control the handshake with a wait signal.

For a typical read or write operation, the replying agent must be aware of how much time it requires (in bus clocks) to send or receive the information on the bus. As an example, an agent may require two bus clocks of extra time for each bus operation. To make the requesting agent extend the operation by two bus cycles, the replying agent holds the wait signal active on the bus for a period of two bus clocks.

The wait signal provides the replying agents with a certain degree of control over the cycles on the bus. During the reply phase, the replying agent uses the wait signal to delay transfer of data on the bus. The wait signal freezes the state of the transfer cycle and the exception cycle on the bus.

If the requesting agent is engaged in a pipeline operation, a replying agent may assert the wait signal during the first clock of the request phase of the next bus operation. Under these circumstances, the replying agent causes the requesting agent to stretch the request phase for as long as the wait signal is valid.

The arbitration cycle is not affected by the wait signal. During the handshake, the primary and secondary requesting agents may continue normal arbitration cycle sequences.

#### 1.5.7 Bus Cycles On The iLBX™ II Bus

The iLBX II bus operates in defined bus cycles, as shown in Figure 1-4. The three types of bus cycles are arbitration cycle, transfer cycle, and exception cycle. Each is described in the following paragraphs.

**1.5.7.1 ARBITRATION CYCLE.** The arbitration cycle ensures that one and only one requesting agent initiates a transfer cycle on the iLBX II bus at a given time. When the secondary requesting agent requires bus access, the agent enters an arbitration cycle to request the use of the bus.

The iLBX II bus allows only two requesting agents, the primary and the secondary. For either agent, the arbitration cycle lasts until the agent acquires access rights to the bus.

In configurations with only a primary requesting agent, the arbitration cycle is eliminated; the primary agent always has immediate access to the bus. In configurations with both primary and secondary requesting agents, the primary agent needs to wait for bus access only when the secondary agent is using the bus.

The secondary requesting agent is always required to arbitrate for access to the bus by asserting a bus request signal. The primary requesting agent releases control of the bus to the secondary requesting agent by asserting the bus acknowledge signal.

**1.5.7.2 TRANSFER CYCLE.** The transfer cycle on the iLBX II bus is the means by which agents transfer address, command, and data information on the bus. The transfer cycle consists of two separate phases: the request phase and the reply phase. Every transfer cycle begins with a request phase and ends with a reply phase.

**1.5.7.3 EXCEPTION CYCLE.** The exception cycle on the iLBX II bus allows bus agents to identify and report error conditions that occur during a transfer cycle. ECC errors, parity errors, and continuation (boundary crossing) errors are examples of typical exception errors for which replying agents start exception cycles.

Exception cycles overlap the request and reply phases of transfer cycles in which they are sensed, overlap arbitration cycles and overlap the request phase of the following transfer cycle.

The bus protocol explicitly defines the timing for exception reporting, however, the protocol has no provisions for recovering from exceptions. This implies that exceptions do not terminate bus cycles. The agents involved in an operation containing an exception may recover from the exception condition in subsequent bus cycles.

The bus protocol provides a mechanism for reporting three different exceptions: the address error, the data error, and the continuation error.

Table 3-1. Signal Notation Summary

Signal Name	Electrical Notation	Logical Notation	State
XBUSREQ	H, TTL high	1 or true	Active, Asserted
	L, TTL low	0 or false	Inactive, Deasserted
XBUSACK	L, TTL low	1 or true	Active, Asserted
	H, TTL high	0 or false	Inactive, Deasserted

new signals on the iLBX II bus are more easily or conveniently discussed as a group. Names for these signals follow a decimal radix numbering convention. Within each signal group, the least significant bit of the group has the suffix '0' following the group name. Successively higher order bits are given higher decimal number suffixes. As an example, XA0 through XA30 refers to the 32 address signal lines.



## iLBX™ II BUS SPECIFICATION

### 2. SIGNAL DESCRIPTIONS

#### 2.1 GENERAL

This section of the specification lists the signal groups, names each signal on the iLBX II bus, and describes the functions of each signal. The signals on the iLBX II bus are presented in four groups: the arbitration cycle signal group, the transfer cycle signal group, the exception cycle signal group, and the system control signal group.

#### 2.2 SIGNAL NAMING AND NOTATIONAL CONVENTIONS

All iLBX II bus signal mnemonics start with a letter "X" to distinguish them from similar signals in other parts of the Multibus II Bus Architecture Specification.

This section of the text uses a consistent method for showing active states of signals. The terms one:zero and true:false can be ambiguous, so their use is avoided. The text uses the terms electrical high and low (H and L) to indicate the state of a signal. An asterisk following a signal name indicates that the signal is active when low. Table 2-1 explains the signal naming notation used in this specification.

Table 2-1. Signal Notation Summary

Signal Name	Electrical Notation	Logical Notation	State
XBUSREQ	H, TTL high	1 or True	Active, Asserted
	L, TTL low	0 or False	Inactive, Deasserted
XBUSREQ*	L, TTL low	1 or True	Active, Asserted
	H, TTL high	0 or False	Inactive, Deasserted

Many signals on the iLBX II bus are more easily or conveniently discussed as a group. Names for these signals follow a decimal radix numbering convention. Within each signal group, the least significant bit of the group has the suffix '0' following the group name. Successively higher order bits are given higher decimal number suffixes. As an example, XA25 through XA00 refers to the 26 address signal lines.

## ILBX™ II BUS SPECIFICATION

### 2.3 SIGNAL GROUPS

The iLBX II bus contains four groups of signals over which requesting and replying agents can enact the protocol. The signal groups and their functions are listed in Table 2-2.

Table 2-2. Signal Groups On The iLBX™ II Bus

Signal Group Name	Description
Arbitration Cycle Signal Group	Provides the bus requesting and bus granting signals and identifies the current owner of the bus.
Transfer Cycle Signal Group	Provides the control, address, data, and parity signals for read and write operations during transfer cycles.
Exception Cycle Signal Group	Provides the error detection signals to show that a problem occurred during a transfer cycle.
System Control Signal Group	Provides system-wide services such as reset and initialization control.

#### 2.3.1 Arbitration Cycle Signal Group

The arbitration cycle signal group consists of two signals: bus request (XBUSREQ\*) and bus acknowledge (XBUSACK\*). The two signals allow the two requesting agents, the primary and the secondary, to perform a bilateral handshake while exchanging ownership of the bus. The condition of the signals also identifies which of the two requesting agents owns control of the bus at any given instant.

**2.3.1.1 BUS REQUEST SIGNAL (XBUSREQ\*).** The secondary requesting agent asserts the XBUSREQ\* signal to request access to the iLBX II bus. In holding the bus request signal active, the secondary requesting agent attempts to acquire (if not the bus owner) or keep (if the bus owner) ownership of the iLBX II bus. The XBUSREQ\* signal is received by only the primary requesting agent on the bus.

When the secondary requesting agent owns the bus, the agent must hold XBUSREQ\* active for the duration of the operation. This prevents the primary requesting agent from attempting to acquire control of the bus during the secondary agent's operation.

On completing an operation, the secondary requesting agent deactivates the XBUSREQ\* signal, allowing the primary an opportunity to gain control of the bus. The primary requesting agent takes control of the bus on the clock following the deactivation of XBUSREQ\* by deactivating the grant signal (XBUSACK\*).

The operation of the XBUSREQ\* signal by the secondary requesting agent depends on the arbitration policy that the agent implements. The earliest that the secondary requesting agent can deactivate XBUSREQ\* in a transfer cycle is the clock cycle after the reply phase is completed.

**2.3.1.2 BUS ACKNOWLEDGE SIGNAL (XBUSACK\*).** The bus acknowledge signal is driven by only the primary requesting agent; the secondary requesting agent cannot drive the XBUSACK\* signal onto the bus. When asserting XBUSACK\*, the primary tells the secondary requesting agent to assume ownership of the iLBX II bus on the next clock. The signal is received by only the secondary requesting agent on the bus.

The relative timing between the assertion and deassertion of the XBUSREQ\* and XBUSACK\* signals depends on two things:

- o the state of the bus when the secondary asserts the XBUSREQ\* signal.
- o the arbitration algorithm and its implementation within the primary and secondary requesting agents.

The earliest that the primary requesting agent can assert the bus acknowledge signal in response to the bus request (XBUSREQ\*) from the secondary requesting agent during the clock cycle after the completion of its reply phase.

## 2.3.2 Transfer Cycle Signal Group

The transfer cycle signal group consists of 64 signal lines. All requesting agents on the iLBX II bus use the transfer cycle signal group to transfer address/command information to the replying agent(s) during the request phase of a transfer cycle and to transfer data between requesting and replying agents during the reply phase(s).

The transfer cycle signals are divided into two groups, the request phase signals and the reply phase signals. This is important because of the pipelining capability of the bus. At any given instant, the request phase signals may be starting another transfer cycle while the reply phase signals are completing the current transfer cycle.

The request phase signals are the address signals XA25 through XA00, the command signals XC3\* through XC0\*, and the address/command parity signal XAPAR\*. Of these signals, only the use of the address parity signal is optional; the use of all others is required. The reply phase signals are data signals XD31 through XD00, and data parity signal XDPAR\*. Each is described further in the following paragraphs.



## iLBX™ II BUS SPECIFICATION

**2.3.2.1 ADDRESS BUS SIGNALS (XA25 - XA00).** The address bus consists of 26 uni-directional signal lines over which the requesting agent sends the starting physical address of an operation. These lines are driven by the requesting agent which has been granted access to the bus. All replying agents on the bus must receive all address signals.

If the command signals select interconnect space, then the requesting agent drives only the least significant 16 bits of the address with valid address information. All 26 bits of the bus must be driven if parity is supported in the system.

**2.3.2.2 COMMAND BUS SIGNALS (XC3\* - XC0\*).** The command bus consists of four encoded signal lines that define the command. The requesting agent uses the command signals to transfer command information to a replying agent during the request phase of an operation. The command signals identify the type of access that occurs at the address. The requesting agent holds the command lines active only during the request phase.

The XC3\* and XC2\* signals encode the address space and the type of operation. The XC1\* and XC0\* signals specify the data width expected during the bus operation. The command signals are actively driven by the requesting agent simultaneous with the address in a given bus operation.

Table 2-3. iLBX™ II Bus Command Signal Encoding

XC3*		XC2*		XC1*		XC0*	
Address Space		Type Of Access		Data Width During Reply			
H=memory		H=read		H	H=	1 BYTE	
				H	L=	2 BYTES	
L=interconnect		L=write		L	H=	3 BYTES	
				L	L=	4 BYTES	

**2.3.2.3 ADDRESS PARITY SIGNAL (XAPAR\*).** The iLBX II bus protocol does not require parity checking on the address or command bus. However, the iLBX II bus provides a parity signal (XAPAR\*) for generating one parity bit for the 26-bit address bus and the 4-bit command as a compliance level.



## iLBX™ II BUS SPECIFICATION

The parity signal, XAPAR\*, provides even parity for signals XA25 through XA00 and XC3\* through XC0\*. That is, when the sum of the high signal lines on the bus is an even number, then the parity line is not active (XAPAR\* is high). When the sum of the high signal lines on the bus is an odd number, then the parity line is active (XAPAR\* is low).

If any agent in an iLBX II system is checking parity on address/command bus, then all agents must generate correct parity. However, checking parity is optional for all agents. If an agent supports parity generation and checking, then it must be capable of disabling this feature.

**2.3.2.4 DATA BUS SIGNALS (XD31 - XD00).** The data bus consists of 32 bidirectional lines that may be driven by either the requesting agent or the replying agent during the reply phase of a transfer cycle. The data bus signals are positive-true signals on the interface and must be driven and received by all bus agents.

The data bus signals move read data and write data between the requesting and replying agents on the iLBX II bus. The data bus is driven during the reply phase: by the requesting agent in a write operation and by the replying agent in a read operation.

**2.3.2.5 DATA PARITY SIGNAL (XDPAR\*).** This signal provides optional even parity for the 32-bit data bus. That is, when the sum of the high signal lines on the data bus is an even number, the parity line is not active (XDPAR\* is high). When the sum of the high signal lines on the data bus is an odd number, the parity line is active (XDPAR\* is low).

If any agent in the iLBX II system checks parity on the data bus, then all agents interfacing to the bus must be capable of generating correct parity on all data transfers. Checking parity is optional for all agents that support parity.

If an iLBX II system does not support parity, no agent should generate or check parity. If an agent supports parity generation and checking on the data bus, then it must be capable of disabling this feature.

### 2.3.3 Exception Cycle Signal Group

The exception cycle signal group provides the error detection signals to identify a problem during a transfer cycle. The signal group consists of 2 signal lines, data error (XDERR\*) and address error (XAERR\*). Each is described further in the following paragraphs.

**2.3.3.1 DATA ERROR SIGNAL (XDERR\*).** The data error signal is driven active by any iLBX II bus agent that checks the integrity of the data exchange during the reply phase. When an agent asserts the XDERR\* signal, it indicates that an exception condition was sensed during the data transfer on XD31 - XD00. The data exception signal must always be valid during the clock following the data. The cause of the exception can be any condition(s) for which the system may wish to check.

All bus agents that do not check for errors on a data transfer or that do not participate in a data transfer operation must not assert XDERR\*. All agents that must respond to reply phase exceptions (data transfer errors) must monitor the XDERR\* signal on the bus.

**2.3.3.2 ADDRESS ERROR SIGNAL (XAERR\*).** The address error signal is driven active by any iLBX II bus agent that checks the integrity of the address/command exchange during the request phase. All agents that service request phase exceptions must monitor the XAERR\* signal.

The address error signal identifies an error on the transfer of address/command information on the bus. By asserting XAERR\* during the clock cycle following the request phase, an agent indicates that either the address on XA25 - XA00 or the command on XC3\* - XC0\* is in error. The XAERR\* is driven by all agents that check the integrity of the address/command on the bus during the request phase. Only agents that perform integrity checking on the address bus must receive the XAERR\* signal.

When asserted during any clock after the first data transfer, the XAERR\* signal identifies a continuation error on the replying agent during a block transfer operation. Within the block transfer, the continuation error indication occurs on the same clock cycle that the data is supposed to be valid on the bus.

#### 2.3.4 System Control Signal Group

The system control signal group provides system-wide services such as reset, initialization control, bus clock, and access control. The system control signal group consists of 10 signal lines, including XWAIT\*, XACCREQ\*, XLOCK\*, XBTCTL\*, XBCLK\*, XINT\*, XRESET\*, and XID2-XID0. These signals control the interaction of the agents via the arbitration cycle, the transfer cycle, and the exception cycle. Each is described further in the following paragraphs.

**2.3.4.1 WAIT SIGNAL (XWAIT\*).** All replying agents use the XWAIT\* signal to implement the uni-directional handshake to the requesting agent on the iLBX II bus. All agents on the bus must monitor the condition of the XWAIT\* signal. Replying agents use the XWAIT\* signal to inject access delays into the reply phase of a transfer cycle.

When a given requesting agent supports pipelining, the XWAIT\* signal has a second function during the request phase. In a pipelined operation, the reply phase of the previous bus operation is overlapped with the request phase of the current operation. As a result, the XWAIT\* signal asserted in the reply phase of the previous bus operation may be overlapped with the first clock cycle of the request phase for the current bus operation. The assertion of XWAIT\* in the first clock cycle of the request phase requires that the requesting agent extend the address/command and control signals on the bus until XWAIT\* is removed. If XWAIT\* is asserted during the first clock cycle of a request phase, the request phase must be extended through the clock cycle following removal of XWAIT\*.

During the reply phase, the replying agent can assert XWAIT\* to give a not-ready-for-data-transfer indication. As a result of activating XWAIT\*, the replying agent extends the reply phase of the transfer cycle. Only the replying agent addressed by the request phase is allowed to use XWAIT\* to extend the reply phase of the transfer cycle. All other replying agents must avoid asserting XWAIT\* until the reply phase of the next transfer cycle.

During reset/initialization, all agents assert the XWAIT\* signal until they have completed their reset/initialization sequence. Agents can give themselves more initialization time by asserting XWAIT\* while XRESET\* is active and then holding XWAIT\* asserted until they complete the reset requirements. This implies that agents must drive the XWAIT\* signal at least one clock cycle before XRESET\* is removed. Agents cannot perform bus cycles until both XWAIT\* and XRESET\* are removed.

**2.3.4.2 ACCESS REQUEST SIGNAL (XACCREQ\*).** Requesting agent that drives the address/command information onto the bus asserts the access request signal to indicate to replying agents that two conditions are true on the bus:

- o the request phase of the transfer cycle is occurring.
- o the address and command information is valid on the XA25 - XA00 and the XC3\* - XC0\* signal lines on the current and following clock cycles.

The requesting agent drives the XACCREQ\* signal active until the clock cycle before the end of the request phase. The XACCREQ\* signal identifies both the beginning and the end of the request phase of the transfer cycle for all agents. Its assertion signals the beginning of the request phase and its deassertion signals that the end of the request phase will occur on the following clock cycle.

If the XWAIT\* signal is asserted in the first clock of a request phase (while the requesting agent asserts XACCREQ\*), then the requesting agent must hold the XACCREQ\* signal active until the cycle following the deactivation of the XWAIT\* signal.



2.3.4.3 LOCK SIGNAL (XLOCK\*). The lock signal is the mechanism through which a requesting agent enforces mutual exclusion on the iLBX II bus. A requesting agent that performs non-divisible bus operations to one or more replying agents on the bus must drive the XLOCK\* signal active. All replying agents on the iLBX II bus must monitor the condition of the XLOCK\* signal. The XLOCK\* signal performs the same function during both the request phase and the reply phase of a transfer cycle.

When a requesting agent asserts the XLOCK\* signal, it causes the replying agent (that is involved in the current bus operation) to lock access to its non-iLBX II ports and hold the ports locked until the XLOCK\* signal is removed. The requesting agent holds the XLOCK\* signal active for the duration of the mutually exclusive bus operations. On completing a set of mutually exclusive bus operations, the requesting agent removes the active XLOCK\* signal from the bus. The requesting agent that locks the bus must not release the bus to another requesting agent until it has completed the mutually exclusive bus operations.

2.3.4.4 BLOCK TRANSFER CONTROL SIGNAL (XBTCTL\*). The block transfer control signal (XBTCTL\*) is received by all replying agents and is driven by a requesting agent during a block transfer operation. When asserted, the XBTCTL\* signal tells the replying agents that the current transfer cycle is a block transfer, a bus operation in which one request phase requires two or more data transfer periods during the reply phase.

The requesting agent asserts the XBTCTL\* signal in the last clock of the request phase of a transfer cycle (the first clock cycle of the reply phase). During a block transfer, the requesting agent holds the XBTCTL\* signal active.

At the end of the block transfer, the requesting agent removes the signal during the clock cycle in which the last data transfer period of the reply phase is expected. When a requesting agent deactivates the XBTCTL\* signal, it tells the replying agent(s) that the data transfer currently in progress is the last of the current block operation.

If during the cycle that the requesting agent asserts or deasserts the XBTCTL\* signal and the replying agent asserts the XWAIT\* signal, then the requesting agent must continue asserting or deasserting the signal on the following clock, until the replying agent removes the XWAIT\* signal.

2.3.4.5 INTERRUPT SIGNAL (XINT\*). This signal is received by only the primary requesting agent. It can be driven by the secondary requesting agent or any replying agent in the system. The XINT\* signal provides non-bus-vectored interrupt capability between the replying agents/secondary requesting agent and the primary requesting agent. A non-bus-vectored interrupt must be completely handled by the primary requesting agent; no vector address information is transferred on the iLBX II bus.



2.3.4.6 BUS CLOCK SIGNAL (XBCLK\*). The XBCLK\* signal provides the reference timing for all synchronous operations on the iLBX II bus. The primary requesting agent must be the only agent in the system that provides an XBCLK\*. All bus agents receive XBCLK\* and synchronize their bus activities with the falling edge of the clock. The frequency of XBCLK\* is intended to be related to the transfer rate clock of the primary requesting agent.

2.3.4.7 RESET SIGNAL (XRESET\*). This signal is driven by the primary requesting agent and received by all other agents interfacing to the iLBX II bus. The primary requesting agent uses XRESET\* to initialize the iLBX II bus environment. The extent of initialization performed by asserting XRESET\* is determined by the system design requirements and bounded only by requirements imposed by the Parallel System bus in the Multibus II bus architecture.

2.3.4.8 IDENTIFICATION SIGNALS (XID2 - XID0). These signals are the board identification lines that contain the three least significant bits of the interconnect address space for the cardslot ID field. The XID2 - XID0 signals are not bussed across the backplane. Rather, they are directly connected to the ground plane or not connected on each cardslot of the backplane. All agents must connect the XID signals to +5 volts via a 10K ohm pull-up resistor. This provides a unique ID for each cardslot. The primary requesting agent's cardslot is assigned an ID of 0. The other slots are assigned IDs of 1 through 5.

The upper two bits of the cardslot ID for interconnect address space are connected to TTL high for accesses to interconnect space on an iLBX II bus agent.

2.3.4.9 POWER SIGNALS (+5V AND GROUND). The iLBX II bus only provides for +5 volt DC and ground; no other power supplies are bussed. The iLBX II bus contains six signal pins assigned to +5V DC on every agent. There are also eight ground pins on the bus (includes one XIDx pin).

## 3. iLBX™ II BUS PROTOCOL

## 3.1 GENERAL

This section uses the signals presented in section 2 to describe the protocol on the iLBX II bus. The description of the protocol is organized according to the three cycles that occur on the bus: the arbitration cycle, the transfer cycle, and the exception cycle. The protocol for each cycle is presented in two stages, first in timing diagrams and then in state-flow diagrams.

The timing diagrams provide a time-ordered representation of bus operations by showing the transition relationships among the signals that create the bus protocol. The timing diagrams illustrate the synchronous nature of the iLBX II bus by showing the bus clocks as vertical lines. The bus clock provides to all agents a common signal for synchronizing operations.

State-flow diagrams present the lowest level of detail in defining the protocol responsibilities of each agent in an operation. The state-flow diagrams list signal conditions required for each state-transition in a cycle. Requesting agents initiate arbitration cycles, transfer cycles, and exception cycles, and replying agents can respond to transfer cycles, initiate exception cycles, and extend cycles on the bus.

The three types of bus cycles and the active signals that create the protocol for each cycle are as follows:

Arbitration Cycle	- XBUSREQ*	Bus Request Signal
	- XBUSACK*	Bus Acknowledge Signal
Transfer Cycle	- XA25 through XA00	Address Lines
	- XC3* through XC0*	Command Lines
	- XD31 through XD00	Data Lines
	- XAPAR*	Address Parity
	- XDPAR*	Data Parity
Exception Cycle	- XAERR*	Address Exception
	- XDERR*	Data Exception

In addition to these basic signals, the requesting and replying agents rely heavily on other signals in implementing the bus cycles. One that must be described first is the XWAIT\* signal that provides the handshake between the replying and requesting agents.

The arbitration cycle only enters into the protocol when the secondary requesting agent requires control of the bus. If an iLBX II system does not include a secondary requesting agent, then the primary requesting agent does not have to arbitrate for bus access. The transfer cycle consists of a request phase for passing address/command information and a reply phase for passing data information. The exception cycle consists of an error phase for reporting address/command errors and a phase for reporting data errors.

The communication protocol among agents on the iLBX II bus is consistent for all operations. The description of the protocol is presented in four steps. First, the text presents the access-delay and the unidirectional handshake concepts that affect all cycles. Then the text presents each of the three cycles: the arbitration cycle, the transfer cycle, and the exception cycle.

### 3.2 HANDSHAKE PROTOCOL

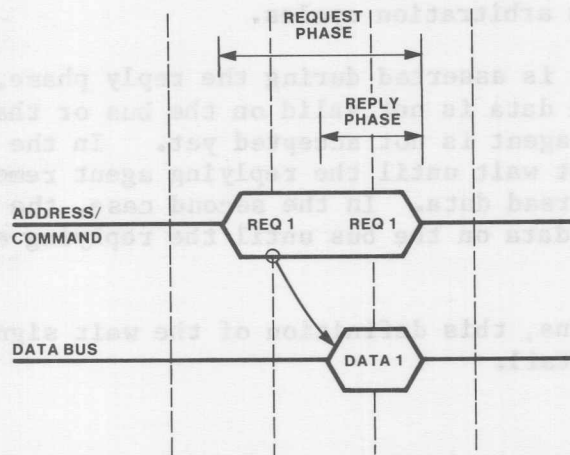
The handshake protocol on the iLBX II bus consists of a combination of the replying agents access delay time (if any) and the replying agents use of the wait signal.

#### 3.2.1 Access Delay Times Of Replying Agents

The iLBX II protocol requires that agents on the bus be tightly coupled. This requires that all replying agents have certain prior knowledge about the speed of the bus (bus clock frequency) and about their own access speeds. Additionally, it requires the requesting agent(s) to be capable of accepting or providing data at each clock.

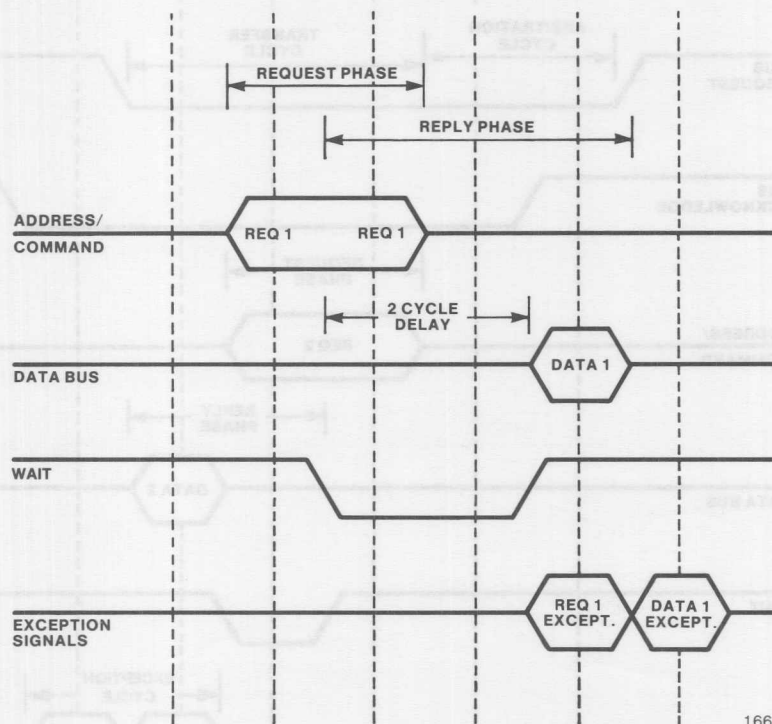
In a best case situation, agents on the bus can operate as shown in Figure 3-1. However, if the replying agent cannot respond quickly enough, it must modify the operation through the use of the wait signal and inject the required number of access delay cycles.

Figure 3-2 shows a diagram of the effects of a two-clock access delay on a single-data transfer cycle; the data is on the bus after the request phase, but may be overlapped with the request phase of the next bus operation if pipelining is supported. Note that the wait signal (XWAIT\*) is asserted in the last clock of the request phase and remains asserted for two clocks to inject a two clock access-delay.



1691

Figure 3-1. No Access Delay Replying Agent Single-Transfer Operation



1664

Figure 3-2. One-Clock Access Delay In A Replying Agent Performing A Single-Transfer Operation



## 3.2.2 Wait Signal Operation

The wait signal (XWAIT\*) provides the replying agents with two functions. The signal can delay transfer cycles and exception cycles, but has no effect on arbitration cycles.

When the wait signal is asserted during the reply phase, it indicates either that the read data is not valid on the bus or that the write data from the requesting agent is not accepted yet. In the first case, the requesting agent must wait until the replying agent removes the signal before sampling the read data. In the second case, the requesting agent must hold the write-data on the bus until the replying agent removes the wait signal.

In subsequent sections, this definition of the wait signal (XWAIT\*) is described in more detail.

## 3.3 iLBX™ II BUS PROTOCOL

The iLBX II bus protocol consists of three bus cycles: the arbitration cycle, the transfer cycle, and the exception cycle. Each of the cycles is shown in Figure 3-3. The figure also shows the two phases of the transfer cycle: the request phase and the reply phase.

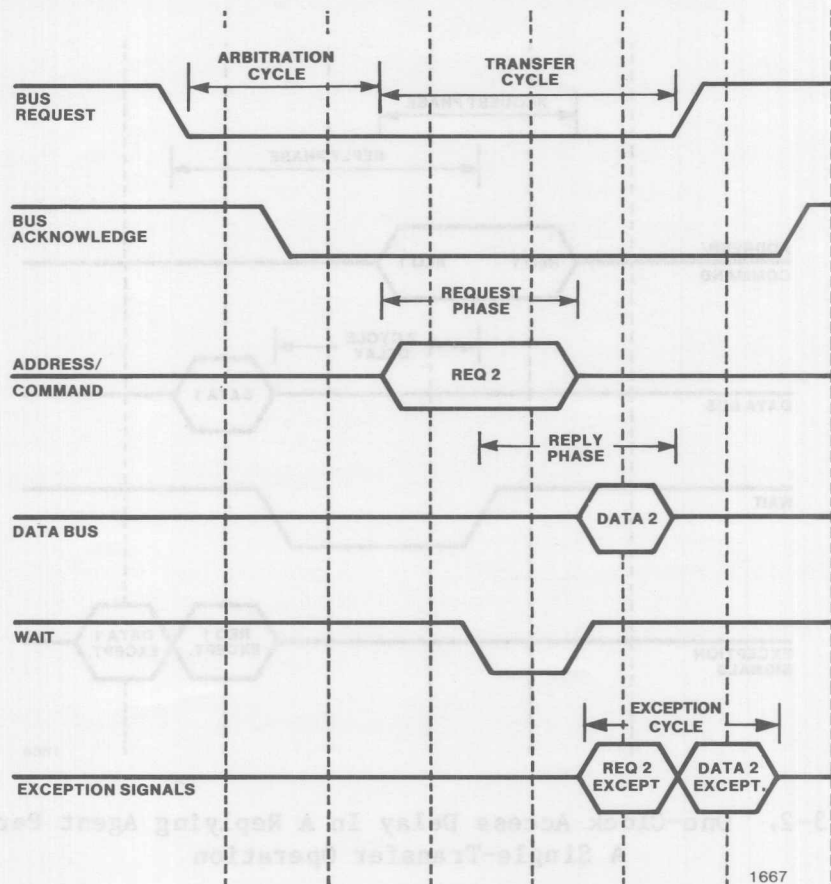


Figure 3-3. Bus Cycles

The arbitration cycle in Figure 3-3 shows a secondary requesting agent obtaining bus ownership from the primary requesting agent. Only the arbitration cycle for the secondary requesting agent is visible on the bus signals. The bus exchange occurs via handshake between the primary and secondary requesting agents: the bus request (XBUSREQ\*) signal from the secondary requesting agent and the bus acknowledge (XBUSACK\*) signal from the primary requesting agent.

The transfer cycle follows one clock after the arbitration cycle. The request phase is always two clock cycles except when pipelining is performed. When pipelined bus operations are performed, the request phase may be extended to more than two clock cycles. The reply phase always overlaps the last clock of the request phase and may last for one or more clocks depending on the wait signal and on the length of the reply.

The exception cycle reports two types of exceptions: request phase exceptions and reply phase exceptions. The exception cycle does not terminate the transfer and arbitration cycles, but overlaps each.

The following paragraphs present more details on each of the cycles.

### 3.3.1 ARBITRATION CYCLE

The bus arbitration cycle is the process by which a requesting agent gains control of the bus. The arbitration cycle may overlap either the transfer cycle or the exception cycle. It always injects at least one clock of dead-time on the bus when agents exchange ownership of the bus.

The primary requesting agent must contain the arbitration logic for the iLBX II bus. The arbitration protocol allows users to optimize the arbitration policy to their implementation needs, with some restrictions.

A maximum of two requesting agents can interface to the bus. This gives rise to two states of ownership of the bus:

- Primary requesting agent is the bus owner: When the bus is idle or when the primary requesting agent performs an operation on the bus, it is the bus owner. During this time, the secondary requesting agent is either idle or waiting to obtain control of the bus.
- Secondary requesting agent active is the bus owner: The secondary requesting agent enters this state on receiving a bus acknowledge from the requesting agent in response to its bus request. During this time, the primary requesting agent is either idle or waiting to obtain control of the bus.

The primary requesting agent owns control of the bus when it is not in use. The secondary requesting agent makes a request to the primary for use of the bus by asserting the bus request signal (XBUSREQ\*). If both the primary and secondary requesting agents request bus access at the same instant, the primary requesting agent gives itself priority.

When the primary requesting agent is performing a bus operation, it grants access rights to the secondary agent only when the current operation is completed. If the secondary is the only agent requesting access to the bus, then the primary requesting agent grants the use of the bus via the bus acknowledge signal (XBUSACK\*).

The protocol allows for either the primary requesting agent or the secondary requesting agent to retain control of the bus after completing a bus operation. The decision depends on the arbitration algorithm implemented.

Figure 3-4 shows a bus exchange sequence. In the figure, the primary requesting agent and the secondary requesting agent make a simultaneous request for bus access. The primary requesting agent grants itself bus access and enters its request phase, as shown. The REQ1 and REQ2 represent address/command information from the primary and secondary requesting agents, respectively. The request and grant signals for the primary requesting agent are not visible on the bus.

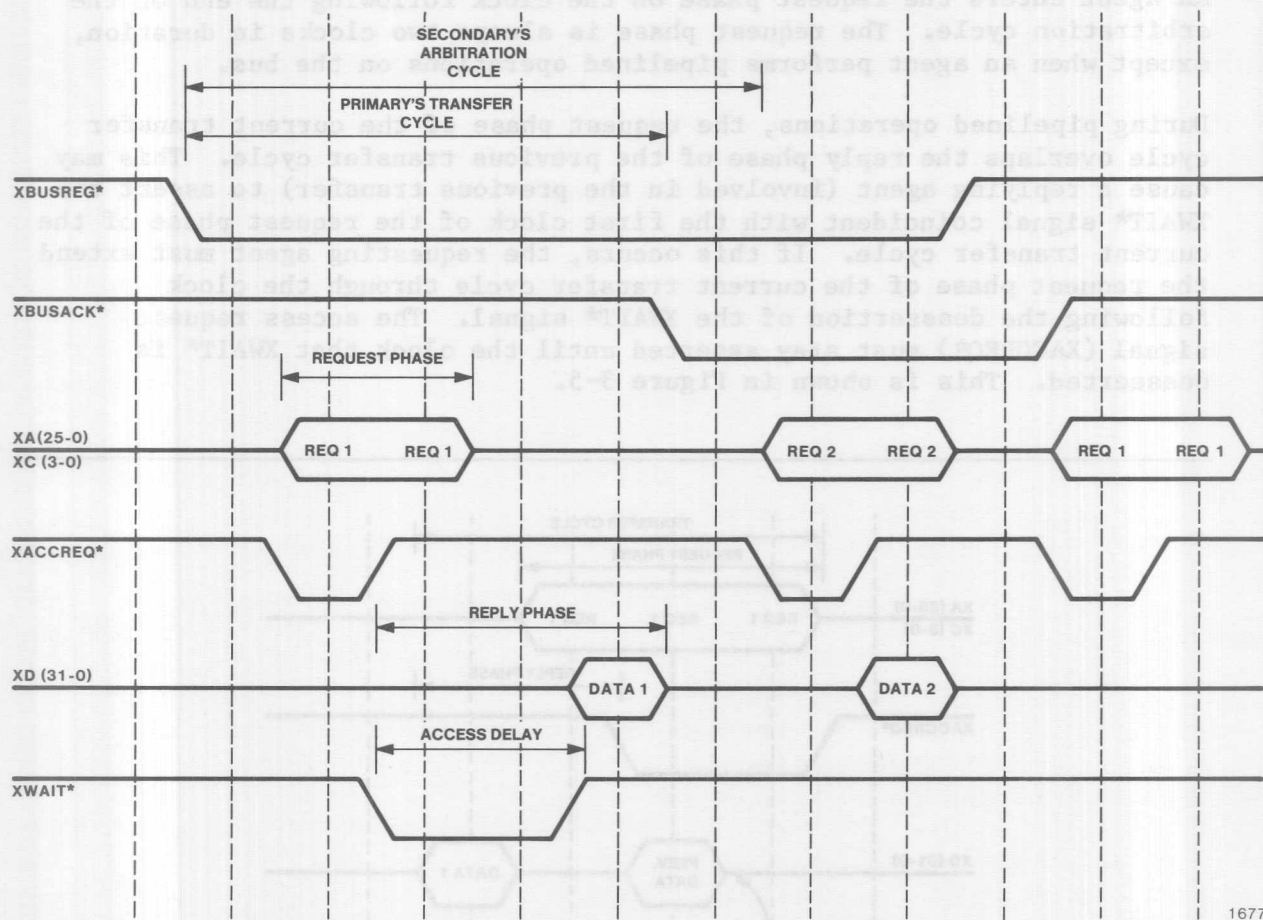
The arbitration cycle for the secondary requesting agent begins at assertion of XBUSREQ\* and continues until the clock cycle following assertion of XBUSACK\* by the primary.

The earliest that the primary requesting agent can issue a bus acknowledge signal in response to the secondary's bus request signal is clock following the completion of its reply phase. Release of the bus to the secondary occurs when the primary asserts XBUSACK\*.

Note that the secondary requesting agent must enter its request phase only if both XBUSACK\* and XBUSREQ\* are asserted during the previous clock.

The arbitration algorithm must allow the secondary requesting agent to complete its bus operation. The primary requesting agent cannot gain ownership of the bus until the secondary removes its XBUSREQ\* signal. This allows the secondary requesting agent to retain control the bus indefinitely, if needed. Although this is allowed by the arbitration algorithm, the secondary requesting agent need not exercise it unless required.

The earliest that the secondary requesting agent can remove XBUSREQ\* in a bus operation is the clock following the completion of its reply phase. On removing its bus request, secondary agent allows the arbitration logic on the primary agent to remove the XBUSACK\* signal and start a transfer cycle on the following clock.



1677

Figure 3-4. Bus Ownership Exchange, Primary-To-Secondary

The arbitration algorithm does not enforce any policy on the secondary requesting agent's control of the XBUSREQ\* signal. It is recommended that the secondary requesting agent be required to release control of the bus by removing its XBUSREQ\* signal for at least one clock after each bus operation. This is not a requirement of the iLBX II bus protocol, however, it will prevent a secondary requesting agent from starving a primary requesting agent.

### 3.3.2 TRANSFER CYCLE

After an agent gains control of the bus (through an arbitration cycle), the agent begins transfer cycles to transfer information to or from a replying agent. Each transfer cycle consists of two phases: the request phase and the reply phase. As Figure 3-5 shows, the transfer cycle begins with the request phase and ends with the reply phase.



## 3.3.2.1 Request Phase Protocol

An agent enters the request phase on the clock following the end of the arbitration cycle. The request phase is always two clocks in duration, except when an agent performs pipelined operations on the bus.

During pipelined operations, the request phase of the current transfer cycle overlaps the reply phase of the previous transfer cycle. This may cause a replying agent (involved in the previous transfer) to assert the XWAIT\* signal coincident with the first clock of the request phase of the current transfer cycle. If this occurs, the requesting agent must extend the request phase of the current transfer cycle through the clock following the deassertion of the XWAIT\* signal. The access request signal (XACCREQ\*) must stay asserted until the clock that XWAIT\* is deasserted. This is shown in Figure 3-5.

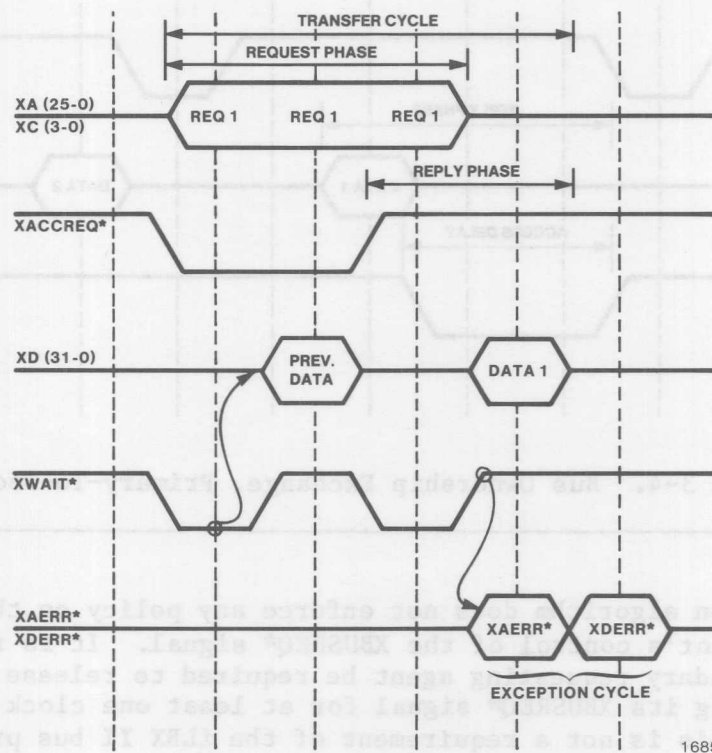


Figure 3-5. Transfer Cycle On The iLBX™ II Bus

If XWAIT\* is not asserted in the first clock of a request phase, then the request phase is always two clock cycles. Assertion of XWAIT\* in the last clock of the request phase delays only the reply phase. The last clock of the request phase always overlaps the first clock of the reply phase.

## iLBX™ II BUS SPECIFICATION

During the request phase, the bus owner places the address and command information onto the address/command bus. During the first clock cycle, the requesting agent also asserts the access request signal (XACCREQ\*) to indicate that the address/command is valid for the current and the following clock. Both the address and command are required to completely define the operation to be performed during the reply phase.

The physical address specifies the physical location of the first byte of the data in the replying agent's address space. The iLBX II bus provides a 26-bit physical byte addressing capability (64 megabytes).

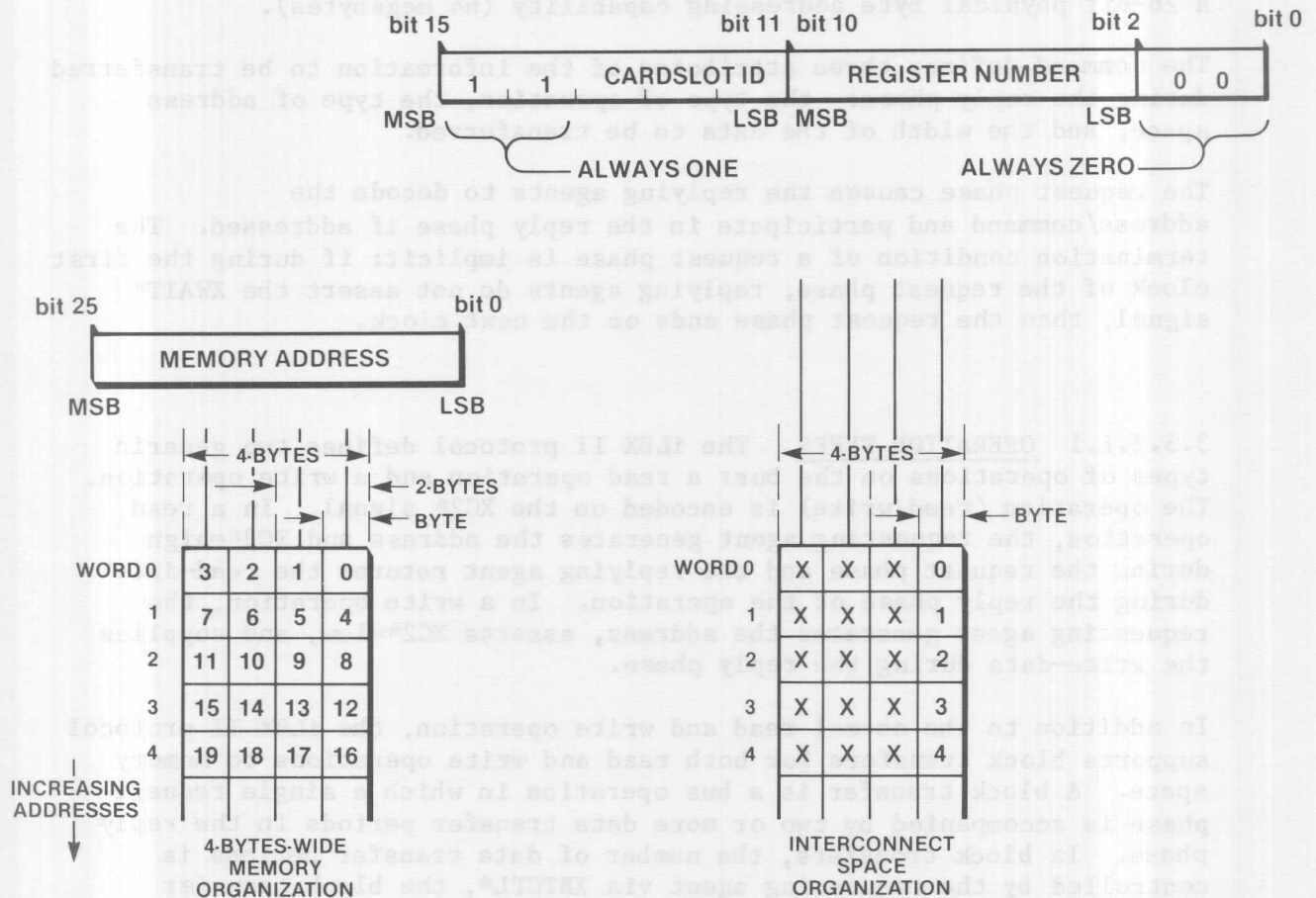
The command defines three attributes of the information to be transferred during the reply phase: the type of operation, the type of address space, and the width of the data to be transferred.

The request phase causes the replying agents to decode the address/command and participate in the reply phase if addressed. The termination condition of a request phase is implicit: if during the first clock of the request phase, replying agents do not assert the XWAIT\* signal, then the request phase ends on the next clock.

**3.3.2.1.1 OPERATION TYPES.** The iLBX II protocol defines two generic types of operations on the bus: a read operation and a write operation. The operation (read/write) is encoded on the XC2\* signal. In a read operation, the requesting agent generates the address and XC2\*=high during the request phase and the replying agent returns the read-data during the reply phase of the operation. In a write operation, the requesting agent generates the address, asserts XC2\*=low, and supplies the write-data during the reply phase.

In addition to the normal read and write operation, the iLBX II protocol supports block transfers for both read and write operations to memory space. A block transfer is a bus operation in which a single request phase is accompanied by two or more data transfer periods in the reply phase. In block transfers, the number of data transfer periods is controlled by the requesting agent via XBTCTL\*, the block transfer control signal. This signal is asserted during the last clock of the request phase and remain asserted until the clock during which the last data transfer period of the reply phase is expected. More details are provided in the reply phase description.

**3.3.2.1.2 ACCESS ADDRESS SPACE.** There are two separate address spaces defined on the iLBX II interface: the memory address space and the interconnect address space. Command bit XC3\* encodes the selection. Either address space can be selected as the target of any access request. Figure 3-6 shows both address spaces, shows their alignment, and shows the address format required for each.



x-630

Figure 3-6. Organization Of Address Spaces  
And Addressing Formats

The memory space has a maximum physical addressing range of 64 megabytes; this translates into a memory address of 26 bits.

The protocol on the iLBX II bus requires that replying agents perform the address incrementing function when performing block transfers in memory space. All data transfers in a block transfer must provide data of a width as specified in the command field, with some restrictions.

## iLBX™ II BUS SPECIFICATION

The interconnect space is used for configuring the iLBX II bus resources on power-up. Each agent on the iLBX II bus has a unique interconnect address set by the iLBX II bus backplane. The interconnect address is always a 16-bit address with least two significant bits set to zero. The remaining fourteen bits consist of a nine bit (512 bytes) interconnect register number and a five bit cardslot identification number. The most significant two bits of the cardslot identification number are always set to a "1", while the least significant three bits provide an encoding of zero through five to identify a given agent on the iLBX II bus. The interconnect space does not support block transfers.

The memory space on the iLBX II bus is required to be organized as an array with a width of (4 times N) bytes, where N is a positive non-zero integer. It must be accessed following a strict alignment policy. The interconnect space is organized as a one byte array but must be addressed in a word (four byte) aligned fashion.

**3.3.2.1.3 DATA WIDTH.** The data width for the data transfer periods of a reply phase are provided by the requesting agent during the request phase. Command bits XC1\* and XC0\* encode the width specification for the transfer.

Agents accessing the memory space may perform operations with 8-, 16-, 24-, and 32-bit data widths. Agents accessing the interconnect space must always require a data width of 8-bits. Since several data widths are allowed, agents must adhere to certain alignment policies for data on the bus. These alignment policies are described in the reply phase description.

### 3.3.2.2 Reply Phase Protocol

During the reply phase of a transfer cycle, the information or data specified in the request phase is transferred between the requesting agent and the replying agent. The first clock of the reply phase overlaps the last clock of the request phase. In a non-block transfer, the reply phase ends on completion of a single data transfer period. In a block transfer, reply phase ends with the clock during which the block transfer control signal (XBTCTL\*) is deasserted and the XWAIT\* signal is not asserted. Figures 3-7, 3-8, and 3-9 show various details about the reply phase of a read, write, and block transfer operation.



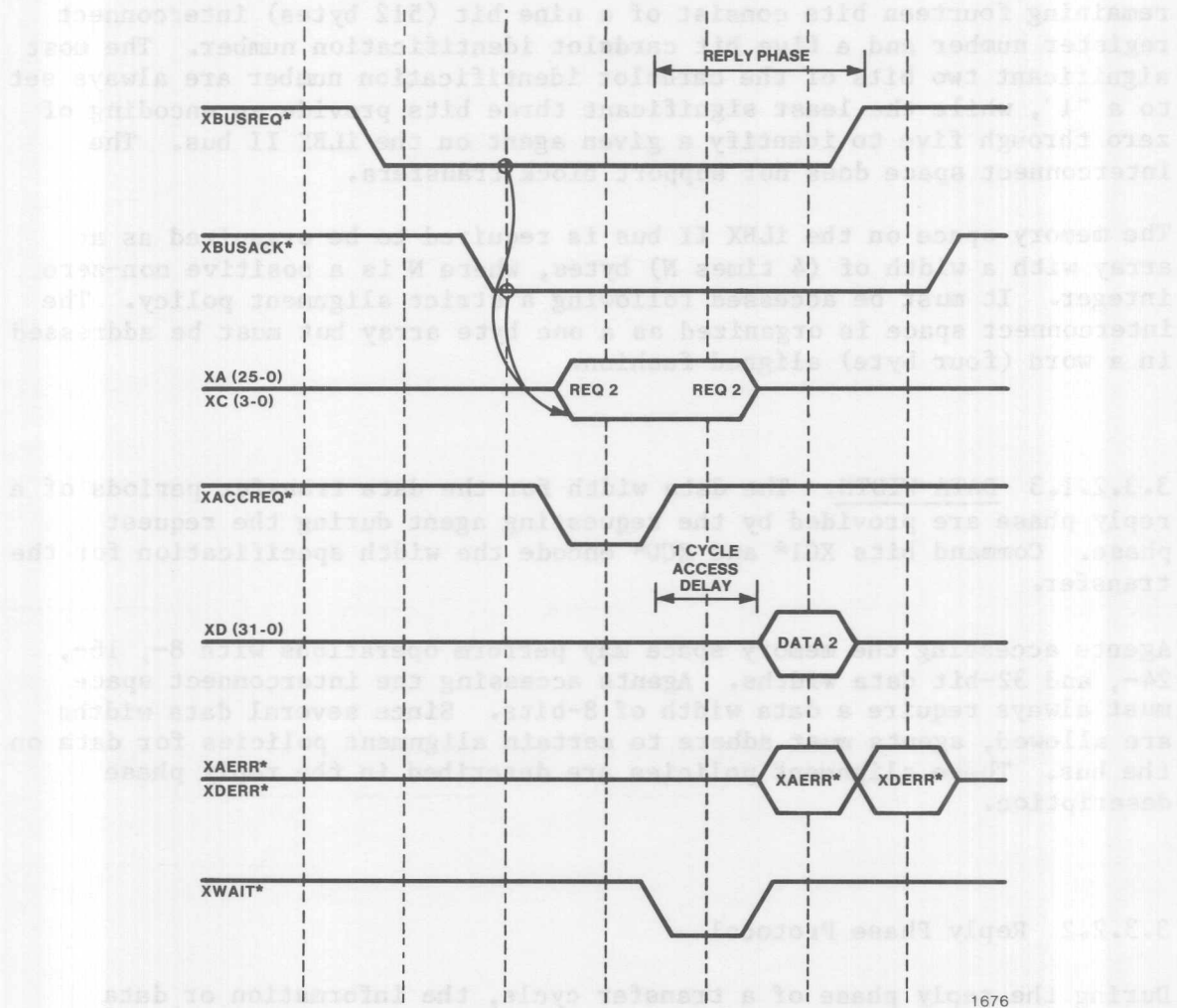


Figure 3-7. READ Operation Reply Phase

# 1LBX™ II BUS SPECIFICATION

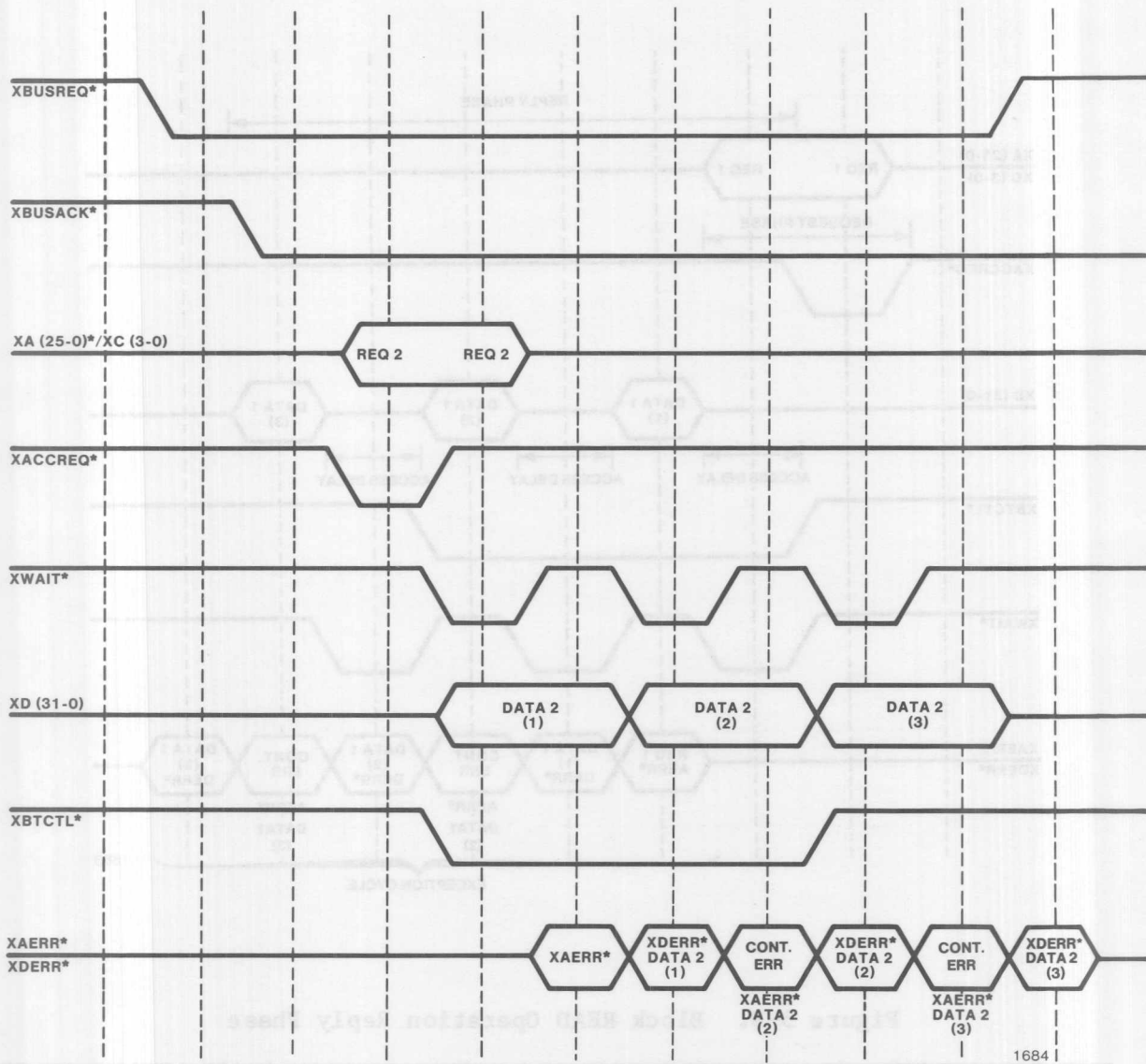


Figure 3-8. Block WRITE Operation Reply Phase

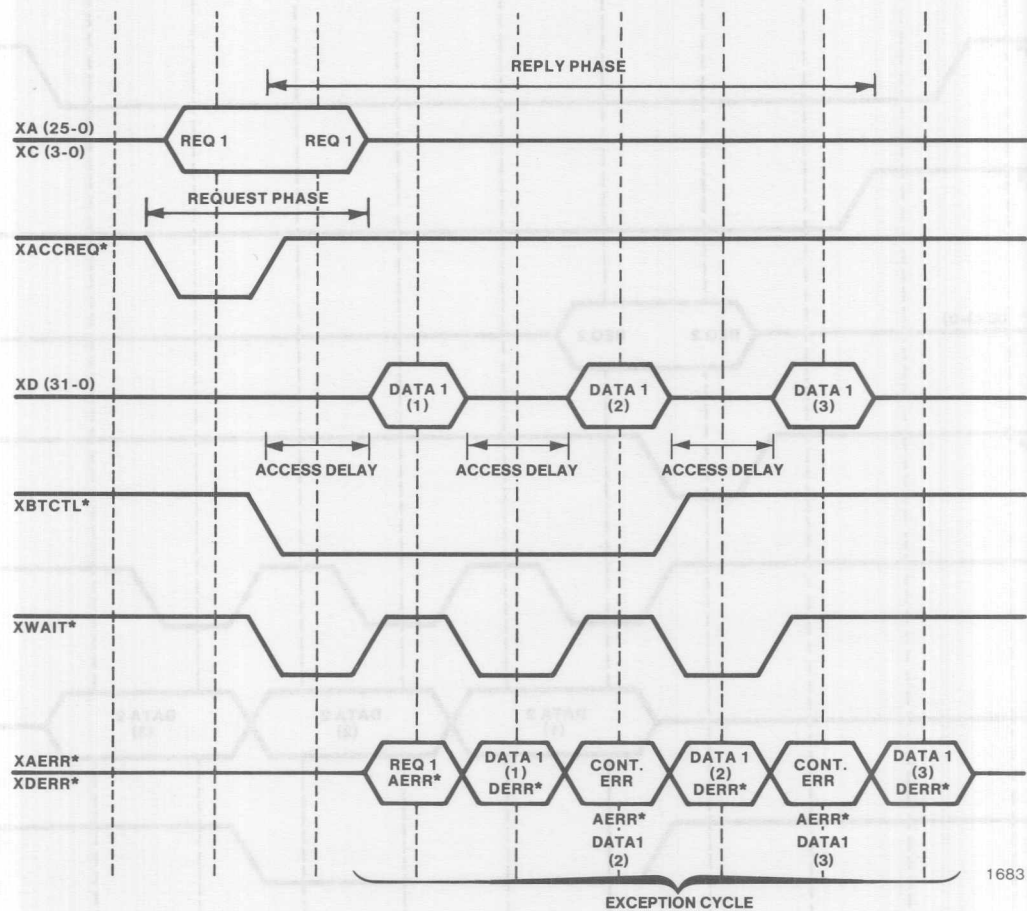


Figure 3-9. Block READ Operation Reply Phase

**3.3.2.2.1 READ OPERATION.** Figure 3-7 shows a secondary requesting agent (REQ2) doing a READ transfer cycle to a replying agent with an access delay of one clock. The access delay is statically programmed on each replying agent. The replying agent asserts XWAIT\* for one clock to obtain the delay needed to perform the bus operation. This access delay is the minimum amount of time required for replying agents to access the first item of data and put it onto the bus.

For a READ operation, the replying agent drives the data onto the appropriate byte-positions of the data bus. It is the responsibility of the requesting agent to pull the appropriate bytes off the data bus depending on the two least significant bits of address and the width specification.

**3.3.2.2.2 WRITE OPERATION.** Figure 3-8 shows a write block transfer operation. In a WRITE operation, the requesting agent drives the write data every clock and extends the data to the following clock if XWAIT\* is asserted. The replying agents are required to extract the appropriate bytes of information from the data bus depending on the two least significant bits of address and the width specification provided in the request phase. This implies that all iLBX II bus agents must have a 32-bit interface to the data bus.

**3.3.2.2.3 BLOCK TRANSFER.** Figure 3-8 and Figure 3-9 show a write and a read block transfer operation, respectively. In block transfers, the reply phase consists of two or more data transfer periods. The requesting agent specifies the starting byte address and the width of the data to be transferred in the request phase. It also controls the length of the reply phase (number of data transfer periods) via the block transfer control signal, XBTCTL\*. Replying agents enter the reply phase of a block transfer the same as during any other access. An important responsibility of the replying agents during a block transfer is to increment the address by the amount specified in the width (XC1\* and XC0\*) for the successive data transfers.

The iLBX II protocol does not support a 24-bit width specification for block transfer operations. The protocol does not allow 16-bit block transfers that are not aligned on 16-bit boundaries.

The requesting agent must assert the block transfer control (XBTCTL\*) signal in the last clock cycle of the request phase. The requesting agent continues driving XBTCTL\* until the clock in which it expects the last data transfer period to start.

The replying agents must acquire their access delay for successive data accesses by asserting the XWAIT\* signal at the appropriate times. The exact time of assertion of XWAIT\* during a block transfer depends on the implementation of the replying agent.



## 3.3.2.3 Data Alignment

An important issue of concern during the reply phase is data alignment on the data bus. The iLBX II bus alignment always requires a 32-bit (word) interface for all agents interfacing to the bus.

Both memory and the interconnect spaces of the iLBX II bus are required to be accessed in a word aligned fashion. This requirement is automatically enforced for the interconnect space based on its restricted addressing and width requirements as shown in Figure 3-6. The memory space must have an array organization whose width is an integer multiple of four bytes. The interconnect space is required to be organized as a four-byte-wide array in which only the low-order byte is valid.

The mapping of the memory and the interconnect spaces to the iLBX II data bus is completely aligned on a 4-byte boundary. This means that byte 0 through 3 of the iLBX II bus are mapped directly to bytes 0 through 3 of the four-byte-wide array in either the memory or the interconnect space. This mapping along with the organization of the two spaces is shown in Figure 3-10.

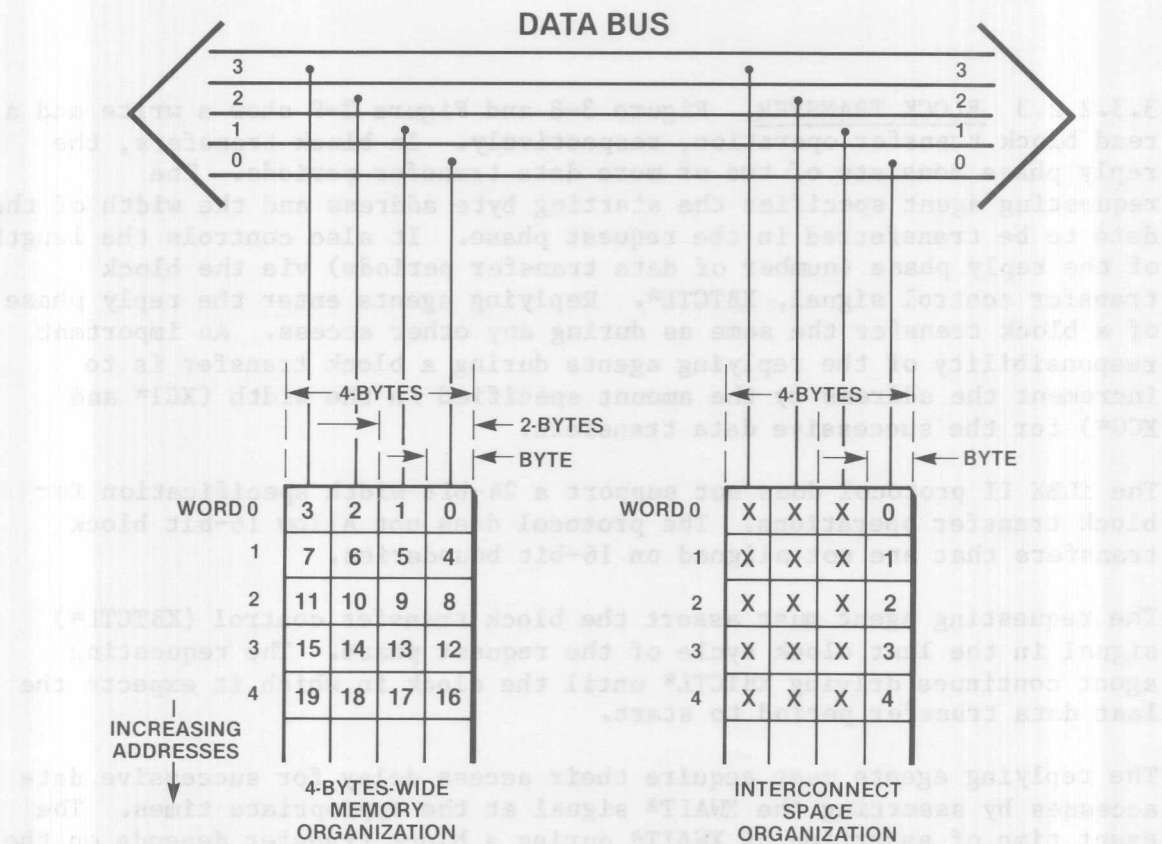


Figure 3-10. Data Alignment And Access Space-to-Bus Mapping

# iLBX™ II BUS SPECIFICATION

Each byte in memory is mapped to one of the four byte locations of the XD31 - XD00 lines. This mapping is determined by the byte's location within a word as illustrated in Figure 3-10.

Based on the alignment policy of iLBX II bus, 8-bit, 16-bit and 32-bit agents must all interface to the bus with a 32-bit interface. Figure 3-11 shows the data alignment interface requirements to the data bus.

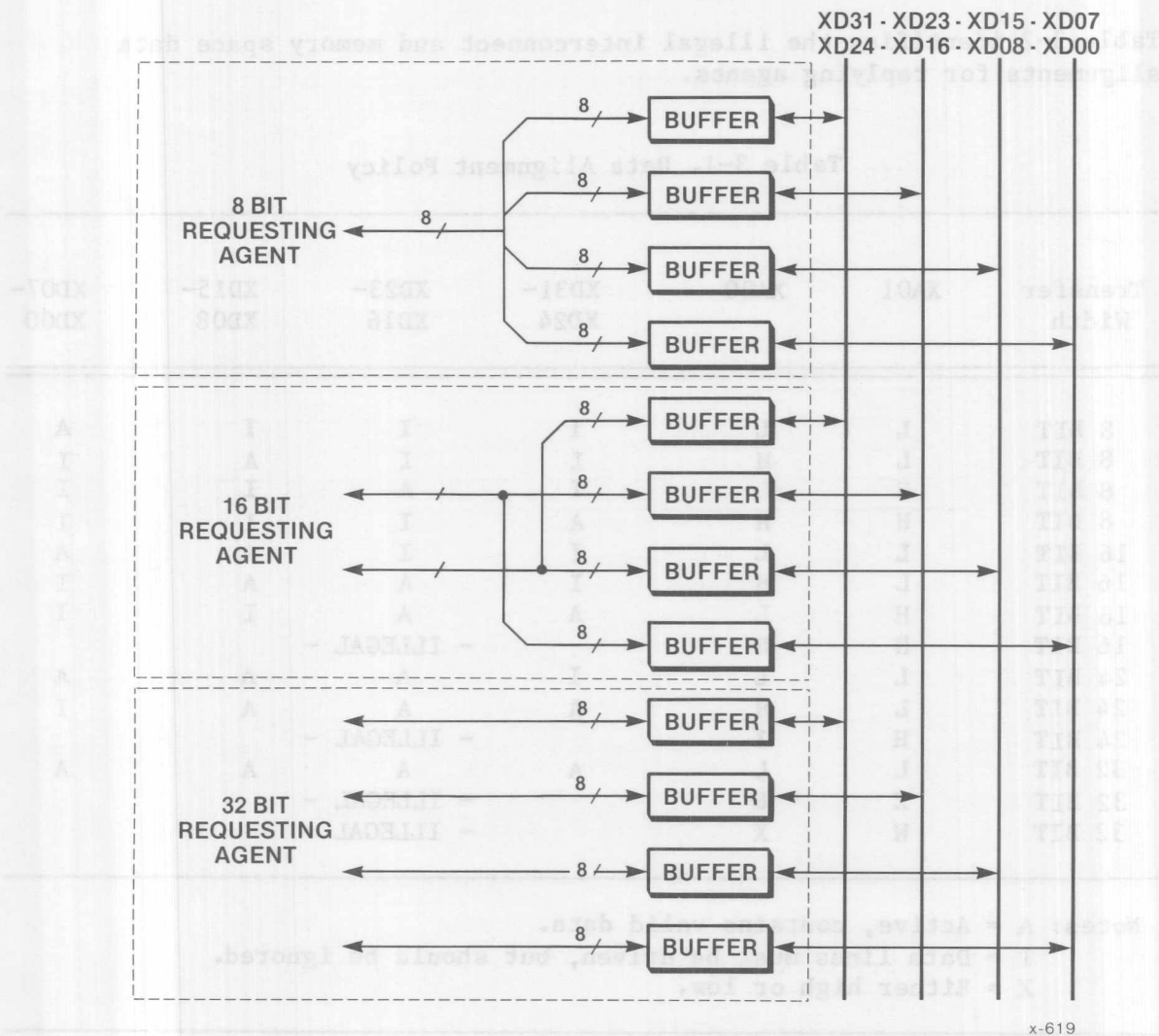


Figure 3-11. Data Alignment Interface Requirements

# iLBX™ II BUS SPECIFICATION

The iLBX II bus alignment policy is performed according to the two least significant address bits and the width specification. As mentioned before, the byte lanes of the data bus correspond to the byte lanes of the replying agents. Both the replying agents and the requesting agents must use the two least significant address bits along with the width to extract the correct information from the data bus during the write and read operations, respectively. This alignment policy requires all iLBX II bus agents to have a 32-bit interface to the data bus.

The information given in Table 3-1 illustrates the alignment policy. As shown, 8-, 16-, 24-, and 32-bit operations are supported by the protocol. However, there are restrictions on the type of the data width specifications for various bus agents.

Table 3-2 identifies the illegal interconnect and memory space data alignments for replying agents.

Table 3-1. Data Alignment Policy

Transfer Width	XA01	XA00	XD31- XD24	XD23- XD16	XD15- XD08	XD07- XD00
8 BIT	L	L	I	I	I	A
8 BIT	L	H	I	I	A	I
8 BIT	H	L	I	A	I	I
8 BIT	H	H	A	I	I	I
16 BIT	L	L	I	I	A	A
16 BIT	L	H	I	A	A	I
16 BIT	H	L	A	A	I	I
16 BIT	H	H		- ILLEGAL -		
24 BIT	L	L	I	A	A	A
24 BIT	L	H	A	A	A	I
24 BIT	H	X		- ILLEGAL -		
32 BIT	L	L	A	A	A	A
32 BIT	X	H		- ILLEGAL -		
32 BIT	H	X		- ILLEGAL -		

Notes: A = Active, contains valid data.

I = Data lines must be driven, but should be ignored.

X = Either high or low.

# iLBX™ II BUS SPECIFICATION

Table 3-2. Address Space Data Alignment Restrictions

Transfer Width	XA01	XA00	Interconnect Space Error	Memory Space Error
8 BIT	L	L	N	N
8 BIT	X	H	Y	N
8 BIT	H	X	Y	N
16 BIT	L	L	Y	N
16 BIT	L	H	Y	N
16 BIT	H	L	Y	N
16 BIT	H	H	Y	Y
24 BIT	L	L	Y	N
24 BIT	L	H	Y	N
24 BIT	H	X	Y	Y
32 BIT	L	L	Y	N
32 BIT	X	H	Y	Y
32 BIT	H	X	Y	Y

Notes: A = Active, contains valid data.

I = Data lines must be driven, but should be ignored.

X = Either high or low.

N = No error, legal access alignment.

Y = Error condition, illegal alignment.

## 3.3.3 EXCEPTION CYCLE PROTOCOL

The iLBX II protocol includes an exception cycle for error reporting capability for all the agents interfacing to the bus. The exception cycle contains both request phase and reply phase exceptions. The iLBX II protocol does not require that requesting and replying agents perform information integrity checking on either the data bus or the address/command bus. Parity is provided as a compliance level. The protocol defines strict timing for reporting of exceptional conditions which may occur during the reply phase or the request phase of a given transfer cycle. The only bus agents allowed to report reply phase exceptions are those engaged in the current bus operation. The request phase exceptions can be reported by all agents.

During the reply phase, the clock following every data transfer period is assigned for reporting any exceptions which might have occurred during that data transfer period. Continuation errors (only applying to block transfer operations) are reported during the same clock in which the associated data appears on the bus. The request phase exceptions must be valid during the first data transfer period.



The iLBX II protocol does not put any requirements on any agent on a response to (or recovery from) an exception. The iLBX II protocol does not allow the requesting agent to extend its ownership of the bus to accommodate any possible recovery needs as a result of sensing an exception. If the recovery involves the use of the bus, the requesting agent must implicitly retain ownership of the bus or re-arbitrate in the form of a new bus request. A replying agent can inject access delays in the transfer cycle to accommodate a possible exception recovery.

Figure 3-12 shows the exception cycle for a read block transfer operation with three data transfer periods. The XAERR\* signal serves two functions: reporting request phase exceptions and reply phase continuation errors. If the XAERR\* signal is asserted during the first data transfer period (DATA1 1), it signals an exception on the request phase information (command/address). If the XAERR\* signal is asserted during the clock following (DATA1 1), it signals a continuation error if XWAIT\* is not asserted. This implies that the continuation errors for the subsequent data transfer periods of a block transfer are reported by only the replying agent participating in the data transfer during the associated data transfer periods.

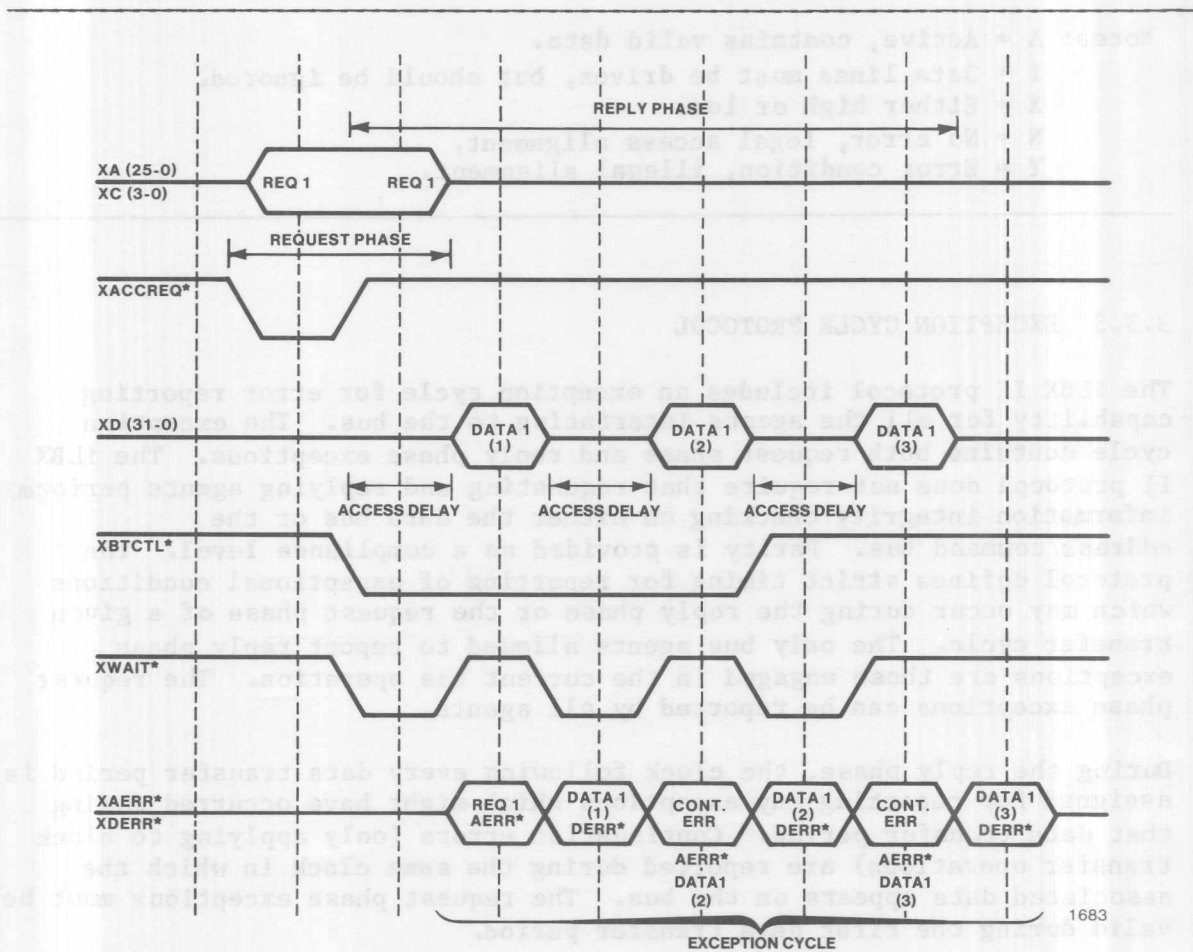


Figure 3-12. Exception Cycle

Exceptions in the data transfer period are always reported in the clock cycle following the data transfer. These exceptions are reported only by agents that participate in the current operation. These exceptions must be reported via the XDERR\* signal.

### 3.4 MUTUAL EXCLUSION

The iLBX II protocol provides a mutual exclusion mechanism (via XLOCK\*) that allows a bus owner to perform a sequence of non-divisible bus operations without other agents performing other bus operations in the interim. Agents typically use mutual exclusion only when performing a sequence of operations that absolutely cannot be interrupted or interleaved with accesses by another agent. The iLBX II bus protocol allows requesting agents to perform mutual exclusion by asserting the XLOCK\* signal during the transfer cycle.

The replying agent that is participating in a transfer cycle and senses an active XLOCK\* signal must prohibit access to its non-iLBX II ports. These ports should remain locked until XLOCK\* is deasserted.

A requesting agent that performs locked transfers must not release the bus to the other requesting agent on iLBX II bus until it removes the XLOCK\* signal.

Figure 3-13 shows a locked transfer operation in which the primary requesting agent performs a read operation followed by a write operation on the bus.

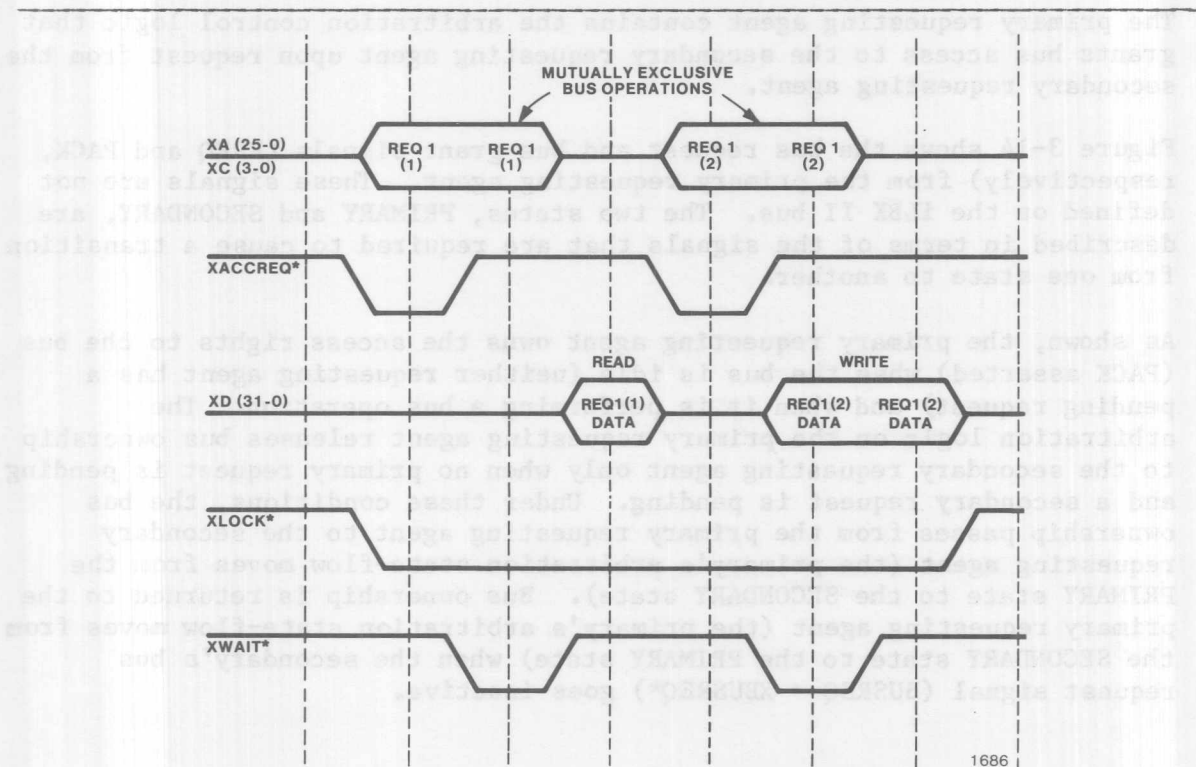


Figure 3-13. Locked Operation

### 3.5 STATE-FLOW DIAGRAMS

The following sections of the iLBX II Bus Specification present the state-flow diagram for the implementation of the bus protocol on the requesting and the replying agents. The state-flow diagrams represent typical control flows for each type of agent. The diagrams are intended to illustrate only the operation of the bus.

#### 3.5.1 Arbitration Cycle

Figure 3-14 shows a state-flow diagram of the arbitration control on the primary requesting agent on the iLBX II bus. As previously stated, a maximum of two requesting agents can interface to the bus. The figure shows two states in which the bus ownership is distinguished:

- Primary requesting agent is the bus owner: When the bus is idle or when the primary requesting agent performs an operation on the bus, it is the bus owner. During this time, the secondary requesting agent is either idle or waiting to obtain control of the bus.
- Secondary requesting agent active is the bus owner: The secondary requesting agent enters this state on receiving a bus acknowledge from the requesting agent in response to its bus request. During this time, the primary requesting agent is either idle or waiting to obtain control of the bus.

The primary requesting agent contains the arbitration control logic that grants bus access to the secondary requesting agent upon request from the secondary requesting agent.

Figure 3-14 shows the bus request and bus grant signals (PREQ and PACK, respectively) from the primary requesting agent. These signals are not defined on the iLBX II bus. The two states, PRIMARY and SECONDARY, are described in terms of the signals that are required to cause a transition from one state to another.

As shown, the primary requesting agent owns the access rights to the bus (PACK asserted) when the bus is idle (neither requesting agent has a pending request) and when it is performing a bus operation. The arbitration logic on the primary requesting agent releases bus ownership to the secondary requesting agent only when no primary request is pending and a secondary request is pending. Under these conditions, the bus ownership passes from the primary requesting agent to the secondary requesting agent (the primary's arbitration state-flow moves from the PRIMARY state to the SECONDARY state). Bus ownership is returned to the primary requesting agent (the primary's arbitration state-flow moves from the SECONDARY state to the PRIMARY state) when the secondary's bus request signal (BUSREQ = XBUSREQ\*) goes inactive.

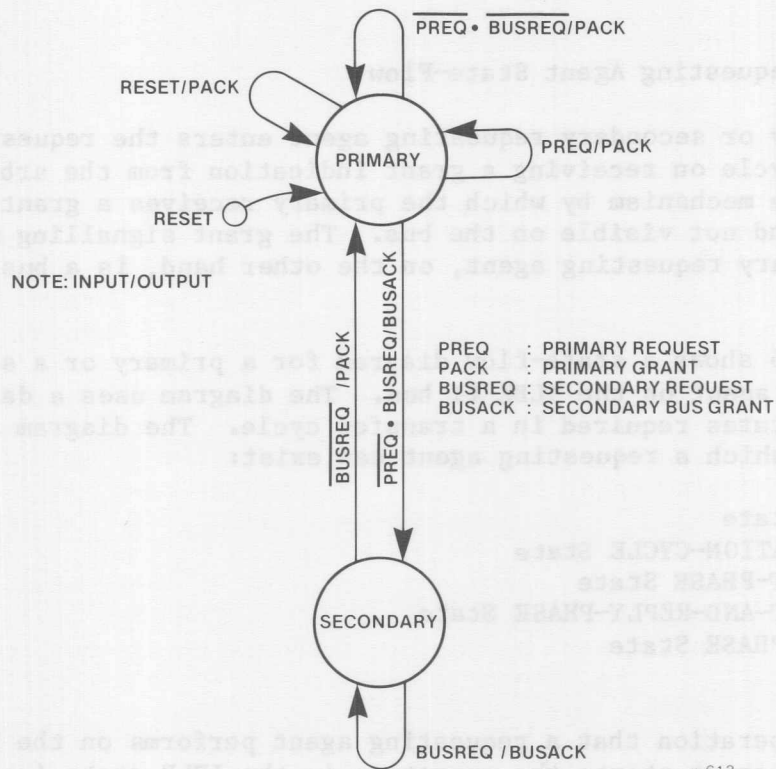


Figure 3-14. State-Flow Diagram For The Arbitration In The Primary Requesting Agent

The bus exchange protocol allows for either the primary requesting agent or the secondary requesting agent to retain control of the bus after completing a bus operation. The decision depends on the arbitration algorithm implemented.

### 3.5.2 Transfer Cycle

During the transfer cycle, the requesting agent that gains access to the bus drives its address/command information onto the bus during the request phase and a replying agent fulfills the requirements of the transfer in its reply phase. The duration of the transfer cycle depends on the duration of the XWAIT\* signal and on the length of the reply.



The following explanation presents the transfer cycle state-flow for the requesting and replying agents. Figure 3-15 shows the details of the state-flow required in a requesting agent. Figure 3-16 shows the details of the state-flow required in a replying agent.

### 3.5.2.1 Requesting Agent State-Flow

The primary or secondary requesting agent enters the request phase of a transfer cycle on receiving a grant indication from the arbitration logic. The mechanism by which the primary receives a grant signal is implicit and not visible on the bus. The grant signalling mechanism for the secondary requesting agent, on the other hand, is a bus signal called XBUSACK\*.

Figure 3-15 shows a state-flow diagram for a primary or a secondary requesting agent on the iLBX II bus. The diagram uses a dashed box to show the states required in a transfer cycle. The diagram shows five states in which a requesting agent can exist:

- IDLE State
- ARBITRATION-CYCLE State
- REQUEST-PHASE State
- REQUEST-AND-REPLY-PHASE State
- REPLY-PHASE State

For each operation that a requesting agent performs on the bus, the requesting agent starts the operation in the IDLE state (unless doing a pipelined operation). The requesting agent moves to either the ARBITRATION-CYCLE state or the REQUEST-PHASE state, depending on whether or not the agent is the bus owner. The agent begins a transfer cycle only when two conditions are true: the agent makes a request for bus access and is granted bus ownership. This combination of conditions causes the requesting agent to move to REQUEST-PHASE state in Figure 3-15.

In the REQUEST-PHASE state, a requesting agent begins the request phase of a transfer cycle. In this phase, the requesting agent drives the address onto XA25 through XA00, drives the command onto XC3\* through XC0\*, and monitors the condition of the XWAIT\* signal.

A requesting agent spends one clock in the REQUEST-PHASE state if no replying agent asserts the XWAIT\* signal during the first clock of the request phase of a transfer cycle (when pipelined bus operations are performed). If a replying agent asserts XWAIT\* during the first clock of the request phase, then the requesting agent remains in the REQUEST-PHASE state during the first clock and for as many clocks as the XWAIT\* is active.

On reaching the exit condition from the REQUEST-PHASE state, a requesting agent enters the REQUEST-AND-REPLY-PHASE state. In this state, the agent continues the request phase of the transfer cycle and begins the reply phase of the transfer cycle. If the reply phase requires only one data transfer period, then the DONE indication causes a transition from the

REQUEST-AND-REPLY-PHASE state to the IDLE state if the XWAIT\* signal is not asserted. Otherwise, the requesting agent enters the REPLY-PHASE state. The requesting agent remains in the REPLY-PHASE state until the DONE signal is asserted, indicating that the required number of data transfers is completed.

If a replying agent asserts the XWAIT\* signal while the requesting agent is in this state, then the replying agent delays the data transfer on the bus by as many clocks as XWAIT\* is active.

**BUSREQ:** Bus Request from requesting agent (PREQ from Primary, XBUSREQ\* from Secondary).  
**GRANT:** Arbitration Grant to the Requesting agent (PACK to Primary, XBUSACK\* to Secondary).  
**WAIT:** Unidirectional handshake (XWAIT\* from replying agents).  
**DONE:** Reply length requirements are met (XBTCTL\* from requesting agent).

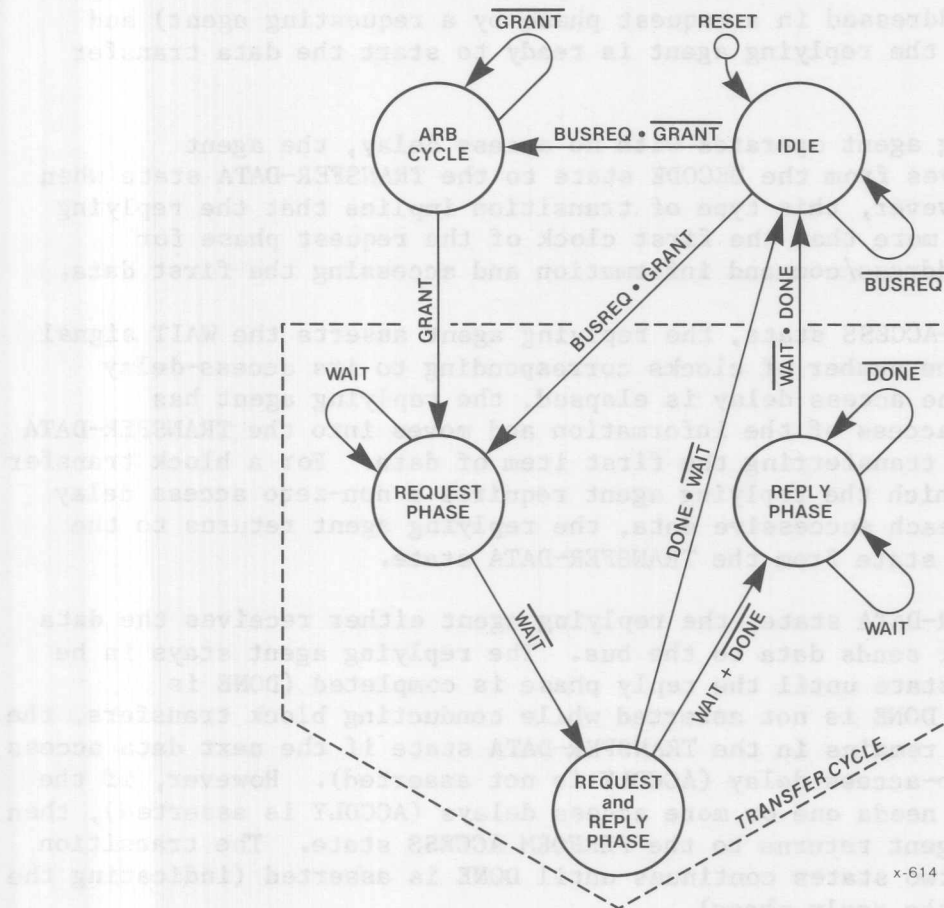


Figure 3-15. State-Flow Diagram For Requesting Agents  
(Primary And Secondary)

## 3.5.2.2 Replying Agent State-Flow

The replying agent is responsible for providing or accepting data in the reply phase. The replying agent can control the progress of the reply phase of a transfer cycle via the XWAIT\* signal.

Figure 3-16 shows the state-flow diagram for a replying agent on the iLBX II bus. The diagram uses a dashed box to identify the request phase and the reply phase of the transfer cycle. The diagram shows three states in which a replying agent can exist:

DECODE State

PERFORM-ACCESS State

TRANSFER-DATA State

All replying agents are always in the decode state and checking for a request phase (XACCREQ\* asserted on the bus). On detecting a request phase that addresses the replying agent, the replying agent moves from the DECODE state of Figure 3-16 to either the PERFORM-ACCESS state or the TRANSFER-DATA state. The selection depends on whether or not the agent is selected (addressed in a request phase by a requesting agent) and whether or not the replying agent is ready to start the data transfer operation.

If the replying agent operates with no access delay, the agent immediately moves from the DECODE state to the TRANSFER-DATA state when addressed. However, this type of transition implies that the replying agent needs no more than the first clock of the request phase for decoding the address/command information and accessing the first data.

In the PERFORM-ACCESS state, the replying agent asserts the WAIT signal (XWAIT\*) for the number of clocks corresponding to its access-delay time. After the access delay is elapsed, the replying agent has completed its access of the information and moves into the TRANSFER-DATA state to begin transferring the first item of data. For a block transfer operation in which the replying agent requires a non-zero access delay for accessing each successive data, the replying agent returns to the PERFORM-ACCESS state from the TRANSFER-DATA state.

In the TRANSFER-DATA state, the replying agent either receives the data from the bus or sends data to the bus. The replying agent stays in the TRANSFER-DATA state until the reply phase is completed (DONE is asserted). If DONE is not asserted while conducting block transfers, the replying agent remains in the TRANSFER-DATA state if the next data access requires a zero-access delay (ACCDLY is not asserted). However, if the replying agent needs one or more access delays (ACCDLY is asserted), then the replying agent returns to the PERFORM ACCESS state. The transition between these two states continues until DONE is asserted (indicating the completion of the reply phase).

- ACCREQ: Access request from the requesting agents (XACCREQ\*).
- WAIT: Unidirectional handshake (XWAIT\* from replying agents).
- DONE: Reply length requirements are met (XBTCTL\* from requesting agent).
- ACCDLY: Access delay signal from replying agent, indicating amount of delay required for accessing the next data.

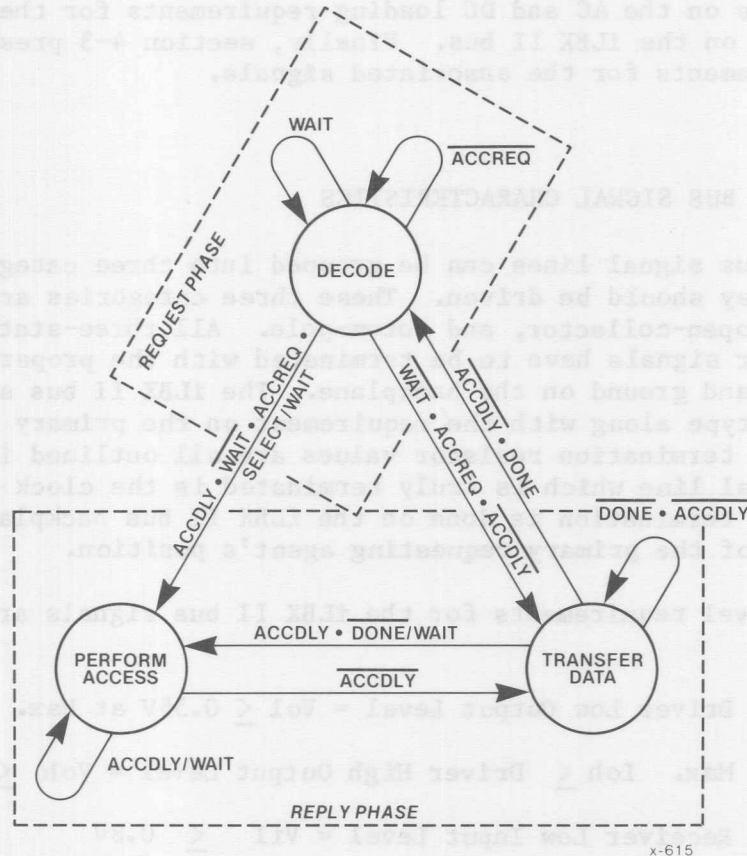


Figure 3-16. Bus Interface State-Flow For A Replying Agent



## 4. ELECTRICAL SPECIFICATIONS

## 4.1 GENERAL

This section addresses three separate topics. Section 4-1 presents a brief summary of the iLBX II bus signal types (three-state, open collector, or totem-pole), pin assignments and the pull-up/pull-down terminations of various signals. Section 4-2 discusses the specifications on the AC and DC loading requirements for the drivers and the receivers on the iLBX II bus. Finally, section 4-3 presents the AC timing requirements for the associated signals.

## 4.2 iLBX™ II BUS SIGNAL CHARACTERISTICS

The iLBX II bus signal lines can be grouped into three categories based on the way they should be driven. These three categories are: three-state, open-collector, and totem-pole. All three-state and open-collector signals have to be terminated with the proper resistor value to +5V and ground on the backplane. The iLBX II bus signals and their driver type along with the requirement on the primary requesting agent for the termination resistor values are all outlined in Table 4-1. The only signal line which is truly terminated is the clock signal line, XBCLK\*. This termination is done on the iLBX II bus backplane at the opposite end of the primary requesting agent's position.

The signal level requirements for the iLBX II bus signals are:

$$0.0V \leq \text{Driver Low Output Level} = V_{ol} \leq 0.55V \text{ at Max. } I_{ol}$$

$$2.4V \text{ at Max. } I_{oh} \leq \text{Driver High Output Level} = V_{oh} \leq 5.0V$$

$$-0.5V \leq \text{Receiver Low Input Level} = V_{il} \leq 0.8V$$

$$2.0V \leq \text{Receiver High Input Level} = V_{ih} \leq 5.25V$$

The iLBX II bus must be implemented on a minimum of four layer printed circuit board backplane using 96-pin DIN 41612 connectors. A minimum of two signal layers is required. The characteristic impedance for an unloaded trace must be in the range of 60 to 90 ohms.

## iLBX™ II BUS SPECIFICATION

Table 4-1. iLBX™ II Bus Signal Type And Terminations

Signal Name	Driver Type	Termination Value
XA(25-0)	three-state	1K ohm to +5, 1K ohm to Ground
XC(3-0)*	three-state	1K ohm to +5, 1K ohm to Ground
XD(31-0)	three-state	150 ohm to +5, 1K ohm to Ground
XAPAR*	three-state	1K ohm to +5, 1K ohm to Ground
XDPAR*	three-state	1K ohm to +5, 1K ohm to Ground
XACCREQ*	three-state	1K ohm to +5, 1K ohm to Ground
XBTCTL*	three-state	1K ohm to +5, 1K ohm to Ground
XLOCK*	three-state	1K ohm to +5, 1K ohm to Ground
XWAIT*	open-collector	110 ohm to +5, 1K ohm to Ground
XINT*	open-collector	110 ohm to +5, 1K ohm to Ground
XDERR*	open-collector	110 ohm to +5, 1K ohm to Ground
XAERR*	open-collector	110 ohm to +5, 1K ohm to Ground
XBUSREQ*	totem-pole	1K ohm to +5, 1K ohm to Ground
XBUSACK*	totem-pole	1K ohm to +5, 1K ohm to Ground
XRESET*	totem-pole	1K ohm to +5, 1K ohm to Ground
XBCLK*	totem-pole	220 ohm to +5, 240 ohm to Ground, both ends of the backplane.
XID(2-0)	no driver	(See note)

Note: XID(2-0) connects to ground at the appropriate pins on the backplane and all agents must pull-up these pins to Vcc via 10K ohm resistors.

All termination resistors are located on the iLBX II bus backplane. The XBCLK\* signal is the only signal that is terminated at both ends of the backplane. Except for the XID2 through XID0, all signals are terminated at one end of the backplane. Termination values are shown in Table 4-1. Note that the three-state signals are required to be driven every clock they must be valid, and no driving board should rely on the pull-ups for the low-to-high transitions.

All signals on the iLBX II bus are bussed across the backplane (except XID2 through XID0). There are no special slot positions for agents. The primary requesting agent must reside in the cardslot with a cardslot ID of 0. The secondary requesting agent must drive the XBUSREQ\* signal and receive the XBUSACK\* signal. The primary requesting agent must drive the XBUSACK\* signal and receive the XBUSREQ\* signal.

# iLBX™ II BUS SPECIFICATION

Table 4-2. iLBX™ II Bus Signal Pin Assignments

Pin No.	Pin Column A	Pin Column B	Pin Column C
1	0 Volts	XC3*	0 Volts
2	+5 Volts	XC2*	+5 Volts
3	Reserved	XC1*	Reserved
4	Reserved	XC0*	Reserved
5	XBTCTL*	XACCREQ*	XLOCK*
6	XAERR*	0 Volts	XWAIT*
7	XA00	XA01	XDERR*
8	XA02	XA03	XA04
9	XA05	XA06	XA07
10	XA08	+5 Volts	XA09
11	XA10	XA11	XA12
12	XA13	XA14	XA15
13	XA16	XID2	XA17
14	XA18	XA19	XA20
15	XA21	XID1	XA22
16	XA23	XA24	XA25
17	XD00	XID0	XAPAR*
18	XD01	XD02	XD03
19	XD04	XD05	XD06
20	XD07	0 Volts	XD08
21	XD09	XD10	XD11
22	XD12	XD13	XD14
23	XD15	XD16	XD17
24	XD18	+5 Volts	XD19
25	XD20	XD21	XD22
26	XD23	XD24	XD25
27	XD26	XD27	XD28
28	XD29	0 Volts	XD30
29	XD31	XDPAR*	XBUSREQ*
30	XRESET*	XINT*	XBUSACK*
31	+5 Volts	Reserved	+5 Volts
32	0 Volts	XBCLK*	0 Volts

Note: The iLBX II bus provides pins to carry 6 amps of +5 volt power and 8 amps at 0 volts.

## iLBX™ II BUS SPECIFICATION

### 4.3 iLBX™ II BUS DC AND AC LOADING SPECIFICATIONS

The AC and DC loading of the iLBX II bus is of paramount importance to correct operation at 12 MHz. This section addresses the constraints that the bus agents interfacing with the iLBX II bus should observe in order to guarantee the operation of the system at 12 MHz.

The distributed capacitive loading of a fully loaded iLBX II bus backplane is restricted to 150 pF per signal trace (there are some exceptions as outlined in Table 4-4). This translates into a total 20 pF of capacitive load per bus agent per signal line and 5 pF for trace and pin capacitance. This capacitive load must be distributed on 0.8 inch minimum spacing across the backplane. As discussed in Section 4.1, all signals are terminated by some resistor value at one end of the backplane except for the XBCLK\* signal, which is terminated at both ends of the backplane via a resistive network of 220 ohms to +5V and 240 ohms to 0 volts.

Table 4-3 shows the necessary electrical requirements for the devices of every bus agent connector interfacing to the iLBX II bus. The devices driving the backplane must meet the requirements given in the first three columns. The first two columns specify the minimum drive (low sink/high source) capability of the driver, and the third column specifies the maximum capacitive load that the device can present to the backplane. The next three columns specify the DC and AC loading requirements for the receiving devices. The fourth and fifth columns of the table specify the maximum low/high input loading of the receivers and the last column is the maximum allowable input capacitance that the receiver can have. If any bus agent connecting to the iLBX II bus has only a receiver or only a driver (not both), then the total device capacitive loading of 20 pF is the maximum allowable input or output capacitance, respectively.

Agents that provide drivers and receivers for a three-state signal on the bus must have the sum of their leakage current in the high impedance state of the driver and the input load of the receiver not exceed the values listed in Table 4-3.

Any device that meets the above specifications can be used to interface to the iLBX II bus backplane lines. The capacitive loading specified is for the current and capacitance measured at the device pins. This does not, however, include the trace capacitance, therefore, care should be taken to make the stub length on the bus agents plugging into the backplane as small as possible. The maximum allowable stub length is 2 inches.



# iLBX™ II BUS SPECIFICATION

Table 4-3. iLBX™ II Bus AC And DC Loading Requirements

Signal Names	Low Drive (mA) Min.	High Drive (mA) Min.	Output Cap. (pFs.) Max.	Iozl plus Iil (mA.) Max. (Note 1)	Iozh plus Iih (uA.) Max. (Note 2)	Input Cap. (pFs.) Max.
XA(25-0)	20	-1	12	-2	150	8
XC(3-0)*	20	-1	12	-2	150	8
XD(31-0)	48	-5	12	-2	500	8
XAPAR*	20	-1	12	-2	150	8
XDPAR*	20	-1	12	-2	150	8
XACCREQ*	20	-1	12	-2	150	8
XBTCTL*	20	-1	12	-2	150	8
XLOCK*	20	-1	12	-2	150	8
XWAIT*(Note 3)	60	O.C.	12	-2	150	8
XDERR*(Note 3)	60	O.C.	12	-2	150	8
XAERR*(Note 3)	60	O.C.	12	-2	150	8
XINT*(Note 3)	60	O.C.	12	-2	150	8
XBUSREQ*	20	-1	12	-2	150	8
XBUSACK*	20	-1	12	-2	150	8
XRESET*	20	-1	12	-2	150	8
XBCLK*	64	-15	12	-2.5	500	8

- Notes:
- For all non-three-state signals, the value of the Iozl is zero and the value of Iil can assume the values specified. The Iil value is the receivers input load at Vin = 0.55 volts.
  - For all non-three-state signals, the value of the Iozh is zero and the value of Iih can assume the values specified. The Iih value is the receivers input load at Vin = 2.40 volts.
  - The high level output leakage of open collector signals must be less than or equal to 400 uA at 5.25 volts.
  - The Iol and Ioh values are as follows:  
Maximum Iol at 0.55 volts  
Maximum Ioh at 2.40 volts

## 4.4 AC TIMING SPECIFICATIONS

All bus agents plugging into the iLBX II bus backplane also have to adhere to certain timing requirements in order to function properly at a 12 MHz clock rate. The iLBX II bus timing parameters are presented in two general categories: the timing requirements for bus agents driving signals, and timing parameters for bus agents receiving signals. In both cases the timing requirements are given with respect to the falling edge of the clock (XBCLK\*) received at the specific bus agent.

## 4.4.1 Signal Driver Requirements

The timing requirements for the bus agents driving data onto the iLBX II bus are specified in terms of two parameters: clock-to-data time and hold time. The clock-to-data time is defined as the maximum amount of time after the clock edge that a driving bus agent must guarantee valid information at its bus interface. This time is calculated by taking into account the temperature and voltage deratings. Hold time is defined as the minimum time after the clock edge that the data remains valid at the driving agent's bus interface. Refer to Figure 4-1.

The exceptions to the above rule are the XACCREQ\*, XDPAR\*, XAPAR\*, XC(3-0)\*, and XD31 - XD00 signals. The driver timing requirements for these signals is specified as a function of the bus clock period (XBCLK\*). At lower frequencies, replying and requesting agents can balance the use of the time-gain in a clock cycle.

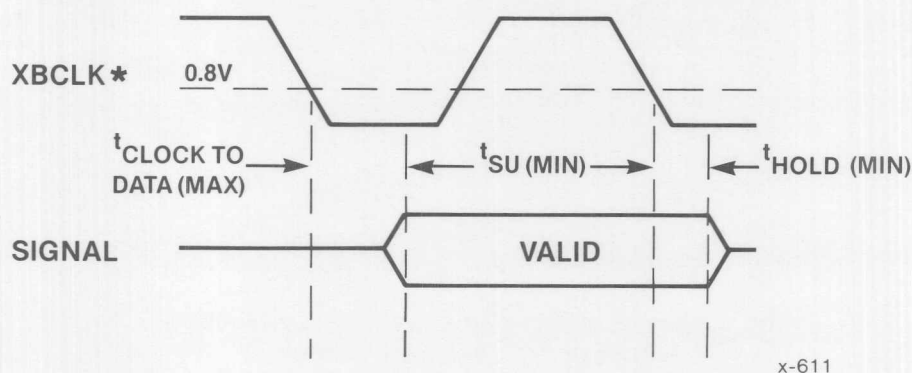


Figure 4-1. Driver Timing Parameter Definition

# iLBX™ II BUS SPECIFICATION

Table 4-4. iLBX™ II Bus Driver Timing Specifications

Signal Name	Clock-To-Data Max. (ns.)	Data Hold Time Min. (ns.)	Cap. Load Max. (pFs.)
XA(25-0)	30	8	150
XC(3-0)*	0.5T-11	8	150
XD(31-0)(read)	0.88T-41	8	150
XD(31-0)(write)	30	8	150
XDPAR*(read)	0.88T-41	8	150
XDPAR*(write)	30	8	150
XAPAR*	30	8	150
XACCREQ*	0.5T	8	150
XBTCTL*	30	8	150
XLOCK*	30	8	150
XAERR*	30	8	150
XDERR*	30	8	150
XWAIT*	35	8	150
XINT*	30	8	150
XBUSREQ*	30	8	50
XBUSACK*	30	8	50
XRESET*	30	8	85
XBCLK*	N.A.	N.A.	85

- Notes: 1. The XBCLK\* period is represented as T.  
2. All of the parameters are rated at 50 pF over the temperature and voltage range.



# iLBX™ II BUS SPECIFICATION

The XBCLK\* signal to which all bus events are synchronized must also meet the following AC characteristics. These characteristics are defined in Figure 4-2.

		Minimum	Maximum	
$t_r$	Rise-time	2	10 ns	Clock rise time from 0.8 to 2.4V
$t_f$	Fall-time	2	7 ns	Clock fall time from 2.4 to 0.8V
	Frequency	1 MHz	12 MHz	Maximum clock frequency
$t_{high}$	High-time	30 ns	T-34	T is the clock period
$t_{low}$	Low-time	30 ns	T-34	T is the clock period

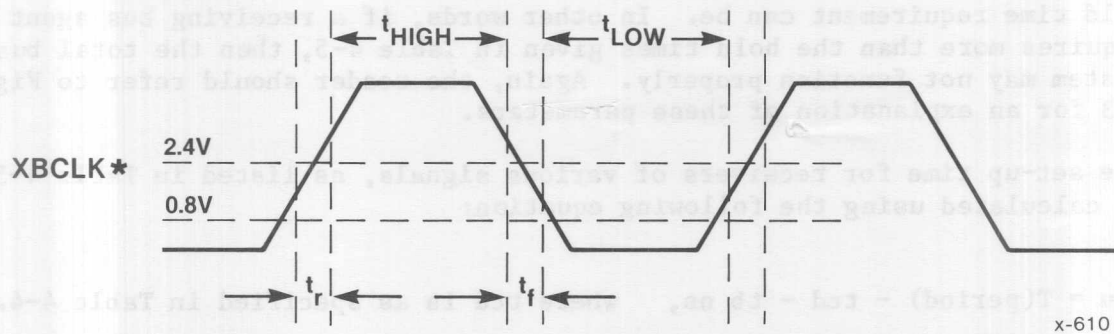


Figure 4-2. Bus Clock Timing Parameters

For three-state signal categories, the signal drivers selected must guarantee that the following is also held true:

maximum	$t_{HOLD}$	<	$t_{CD}$	(Max.)
minimum	$t_{CD}$	≥	$t_{HOLD}$	(Min.)

For the control of cross-talk noise, the maximum slew rate allowed for all signals except the clock is 0.5 volts per nanosecond.



## iLBX™ II BUS SPECIFICATION

### 4.4.2 Signal Receiver Requirements

Timing requirements for the bus agents receiving signals on the iLBX II bus are defined in terms of set-up time and hold time parameters, as defined in Figure 4-3. Every receiving bus agent should only require a minimum set-up and hold time for every input signal which is less than or equal to the minimum values the iLBX II bus can guarantee for 12 MHz operation as specified in Table 4-5.

The minimum set-up time specified is the maximum amount of time that the receivers can assume data is valid prior to the clock edge at which the data is received. This parameter is defined relative to the clock (XBCLK\*) received at the connector edge of the receiving bus agent. See Figure 4-3 for the definition of these parameters.

Data hold time specifies the maximum value that a receiving agent's minimum hold time requirement can be. In other words, if a receiving bus agent requires more than the hold times given in Table 4-5, then the total bus system may not function properly. Again, the reader should refer to Figure 4-3 for an explanation of these parameters.

The set-up time for receivers of various signals, as listed in Table 4-5, is calculated using the following equation:

$$T_{su} = T(\text{period}) - t_{cd} - t_b \text{ ns, where } t_{cd} \text{ is as specified in Table 4-4.}$$

The parameter  $t_b$  is the time required by the bus to transmit the information between any two agents. It includes two propagation delays on the bus and a worst case clock skew. The value of  $t_b$  is referred to as the iLBX II bus loss. This value is 9 ns and is independent of the clock frequency.

The  $t_b$  value is dependent on the  $V_{oh}$ -vs- $I_{oh}$  characteristics and on the  $V_{ol}$ -vs- $I_{ol}$  characteristics of the driver. As a result, the drivers from two TTL families are preferred: the FAST family and the ADVANCED SCHOTTKY family.

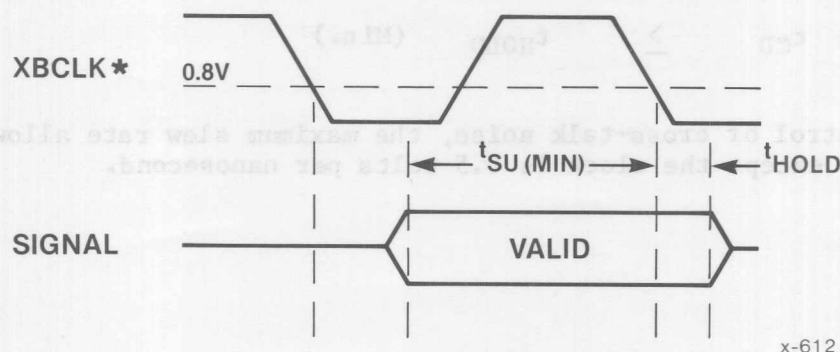


Figure 4-3. iLBX™ II Receiver Timing Parameter Definitions

# iLBX™ II BUS SPECIFICATION

The replying agents must use receiver timing specification as listed in Table 4-5 to calculate their access delay requirements. All bus agents that are not designed to run at 12 MHz must use the AC driver/receiver specifications in Tables 4-4 and 4-5 to identify the maximum frequency at which they operate.

Table 4-5. iLBX™ II Bus Receiver Timing Specifications

Signal Names	Set-Up Time Min. (ns.)	Hold Time Min. (ns.)
XA(25-0)	T-39	4
XC(3-0)*	0.5T+2	4
XD(31-0)(write)	T-39	4
XD(31-0)(read)	0.12T+32	4
XAPAR*	T-39	4
XDPAR*(write)	T-39	4
XDPAR*(read)	0.12T+32	4
XACCREQ*	0.5T-9	4
XBTCTL*	T-39	4
XLOCK*	T-39	4
XWAIT*	T-44	4
XAERR*	T-39	4
XDERR*	T-39	4
XINT*	T-39	4
XBUSREQ*	T-39	4
XBUSACK*	T-39	4
XRESET*	T-39	4
XBCLK*	N.A.	N.A.

## 4.4.3 Preferred Driver/Receiver Components

The iLBX II bus interface does not require drivers that have a high current sink capability. This allows a wider selection of interface components for the bus. The specification relies on the Voh-Ioh and the Vol-Iol characteristics of the drivers in guaranteeing a bus loss (tb) of 9 nanoseconds for open-collector signals.

Any fast or advanced Schottky parts with an Iol and Ioh value that exceeds the values listed in Table 4-3 are suggested as signal drivers for the iLBX II interface.

## 5. COMPLIANCE LEVELS

### 5.1 GENERAL

This section defines the variability allowed within agents on the Parallel System Bus (iPSB bus) portion of the Multibus II System Architecture. The purpose in defining the limits of variability is to assure the maximum amount of upward compatibility. In most cases, agents designed to different levels of compliance create a system with an over-all compliance of the least complex agent.

### 5.2 DATA PATH

The iLBX II bus allows 8-bit, 16-bit, and 32-bit requesting agents to co-exist on the iLBX II bus. However, all replying agents must support 4-byte wide (32-bit) operations in the memory address space. The bus allows consecutive transactions that are directed to different agents of varied bus width. A system implementation that uses agents with more than one bus width for the data path must define the data width of devices properly within the interconnect space for each agent.

### 5.3 ADDRESS PATH

The iLBX II bus a 26-bit address path. Agents on the bus must implement all 26-bits.

### 5.4 COMPLIANCE CODES

The codes assigned to the various areas of compliance for the iLBX II bus are as follows:

- o Type of device - primary or secondary requesting agent PRA
  - secondary requesting agent only SRA
  - replying agent only RA
- o Sequential Transfer Capability - available ST
  - none -
- o Parity Support - on address/command bus PA
  - on data bus PD
  - not supported -

## iLBX™ II BUS SPECIFICATION

### 5.5 COMPLIANCE STATEMENT EXAMPLE

The compliance statement for an iLBX II bus-compatible board should be clearly printed on the board. Omission of any particular compliance code is interpreted as non-support of the capability.

As an example a secondary requesting agent that supports sequential data transfers, supports parity on address/command, and supports parity on data would be marked as follows:

SRA PA PD ST

A replying agent that supports sequential transfers with no parity would be marked as follows:

RPA ST



\*\*\*







# INTEL iSSB™ BUS SPECIFICATION

---





# CONTENTS

	PAGE
SERIAL SYSTEM BUS SPECIFICATION	
1.1 Introduction.....	4-1
1.2 Physical Characteristics.....	4-1
1.3 Access Protocol.....	4-1
1.4 Typical iSSB™ Bus Systems Configurations.....	4-2
1.5 Physical Interface.....	4-3

## TABLE

1-1. iSSB™ Bus Signal Line Encoding.....	4-3
--	-----

## FIGURE

1-1. Typical iSSB™ Bus Systems Configurations.....	4-2
--	-----





PAGE

## ARIAL SYSTEM BUS SPECIFICATION

1.1	Introduction .....	4-1
1.2	Physical Characteristics .....	4-1
1.3	Access Protocol .....	4-1
1.4	Typical 155B Bus System Configurations .....	4-2
1.5	Physical Interface .....	4-3

## TABLE

1-1	155B Bus Signal Line Encoding .....	4-2
-----	-------------------------------------	-----

## FIGURE

1-1	Typical 155B Bus System Configurations .....	4-2
-----	--	-----



## CHAPTER 4 SERIAL BUS SPECIFICATION

### 1.1 INTRODUCTION

At the time of this release, the Serial System (iSSB) Bus Specification is not completed. However, this chapter provides a brief overview of the capabilities and applications for the iSSB bus in a Multibus II system environment.

The iSSB bus portion of the Multibus II bus architecture provides a simple, low cost alternative to the Parallel System Bus (iPSB) message space. By providing the iSSB bus, Intel allows users to easily migrate between the higher performance/higher cost of the iPSB bus and the lower performance/lower cost of the iSSB bus. The iSSB bus serves as a low-cost replacement for the iPSB bus in applications where cost reduction is required and serves as a complement to the iPSB bus where an alternate bus path is required for interface control, diagnostics, or redundancy.

### 1.2 PHYSICAL CHARACTERISTICS

The iSSB bus consists of a maximum of 32 agents (nodes) that may be distributed over a maximum of 10 meters. The agents may be distributed along the cable or clustered into backplanes as shown in Figure 1-1. Each backplane may contain up to 20 agents, the maximum number of cardslots in a Multibus II backplane.

Clustered systems use repeaters as a connection between backplanes and the iSSB bus cable. The repeaters isolate the cable from excessive capacitive load on the backplane.

### 1.3 ACCESS PROTOCOL

The iSSB bus employs access protocol called Carrier-Sense-Multiple-Access with collision detection (CSMA/CD). The CSMA/CD protocol allows agents to transmit data whenever they are ready.

In CSMA/CD operation, an agent looks at the iSSB bus before beginning a transmission. If the bus is not idle, the agent waits until the line becomes idle and until an interframe space has passed. After both events, the agent begins transmission of the message on the iSSB bus.

It is possible for more than one agent to initiate a transmission at the same time; in that case, a collision occurs on the bus. The protocol handles collisions on the iSSB bus via a deterministic collision resolution algorithm that uses time slots.

### iSSB™ BUS TOPOLOGY

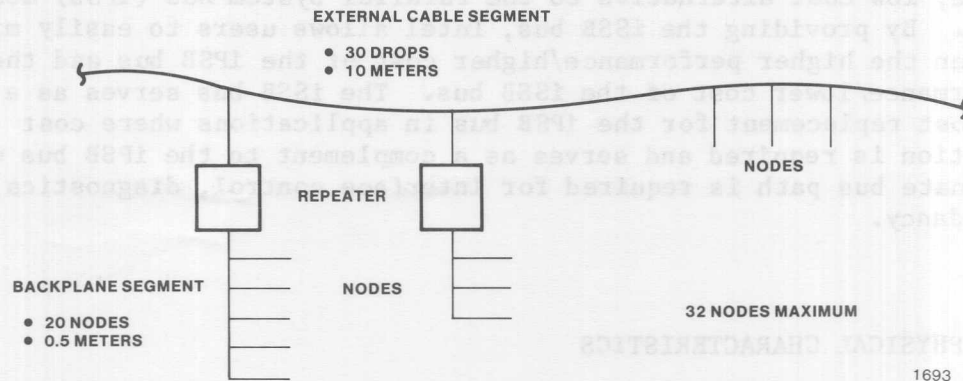


Figure 1-1. Typical iSSB™ Bus Systems Configurations

The deterministic collision resolution algorithm guarantees a time slot during which each agent can gain access without interference from other agents. The resolution grants fair access to all agents. This type of collision resolution allows agents to resolve collisions in a finite time period by providing a real-time response.

#### 1.4 ERROR CONTROL

The iSSB uses a 16 bit CRC for error detection. Used in conjunction with an intelligent interface (to perform retransmission) this allows the iSSB to look as reliable as the iPSB to the application, even though it is up to 10 meters long.

## SERIAL SYSTEM BUS SPECIFICATION

### 1.5 PHYSICAL INTERFACE

The physical part of the iSSB bus interface consists of two signal lines (the SDA and the SDB lines) that are included as part of the iPSB backplane design and may be extended via two signal lines that connect to a repeater, typically located on the CSM. Agents encode data on the complementary, open-collector signal lines as listed in Table 1-1.

The portion of the signal lines within the backplane is designed to operate in a high-noise environment such as a heavily loaded backplane. Cable extensions to the iSSB bus must adhere to normal transmission line requirements.

To further improve reliability, the bus interface includes receivers that sample the data and filter out noise which may be couple from the surrounding environment.

Table 1-1. iSSB™ Bus Signal Line Encoding

SDA Line	SDB Line	Line Condition
0	0	collision
0	1	logic 0
1	0	logic 1
1	1	idle



# SERIAL SYSTEM BUS SPECIFICATION

## 1.3 PHYSICAL INTERFACE

The physical part of the 1228 bus interface consists of two signal lines (the 20A and the 20B lines) that are included as part of the 1228 backplane design and may be extended via two signal lines that connect to a receiver, typically located on the CSM. Agents encode data on the complementary, open-collector signal lines as listed in Table 1-1.

The portion of the signal lines within the backplane is designed to operate in a high-noise environment such as a heavily loaded backplane. Cable extensions to the 1228 bus must adhere to noise transmission line requirements.

To further improve reliability, the bus interface includes receivers that sample the data and filter out noise which may be couple from the surrounding environment.

Table 1-1. 1228 Bus Signal Line Encoding

20A Line	20B Line	Line Condition
0	0	collided
0	1	logic 0
1	0	logic 1
1	1	idle



# **INTEL MULTIBUS® II MECHANICAL SPECIFICATION**

---





# CONTENTS

	PAGE
MECHANICAL SPECIFICATION	
1.1 Scope.....	5-1
1.2 General.....	5-1
1.3 Board Sizes and Dimensions.....	5-3
1.3.1 Printed Board Layout Considerations.....	5-3
1.3.2 Board Mounted Accessories.....	5-3
1.3.3 Board To Board Relationships.....	5-6
1.4 Connectors.....	5-7
1.5 Front Panels.....	5-7
1.6 Backplanes.....	5-11
APPENDIX A	
RECOMMENDED DESIGN PRACTICES	
A.1 General.....	5-14
A.2 Backplane Designs.....	5-14
A.2.1 iLBX™ II Backplane Design.....	5-14
A.2.2 iPSB™ Backplane Design.....	5-18
TABLES	
A-1 iLBX™ II Backplane Characteristics.....	5-15
A-2 Connections Necessary at J5 Connector.....	5-18
A-3 iPSB™ Backplane Characteristics.....	5-19
FIGURES	
1-1. Board Sizes and Bus/Functional Assignment to Connectors....	5-2
1-2. Typical Sub-Rack.....	5-2
1-3. Double-High Board Size.....	5-4
1-4. Single-High Board Size.....	5-5
1-5. Ejector Tab.....	5-6
1-6. Board-to-Board Spacing Relationships.....	5-7
1-7. Board Connector (603-2-IEC-C096M).....	5-8
1-8. Backplane Connector (603-2-IEC-C096F).....	5-8
1-9. Relationship of Mechanical Components.....	5-9
1-10. Double-High Front Panel.....	5-10
1-11. Single-High Front Panel.....	5-11
1-12. Backplane and Mounting Rail Relationship.....	5-12
1-13. Typical Backplane.....	5-13
A-1. Four-Slot iLBX™ II Backplane.....	5-16
A-2. Specifications for the J5 Connector.....	5-17
A-3. Twelve-Slot iPSB™ Backplane.....	A-20
A-4. Twenty-Slot iPSB™ Backplane.....	A-21





PAGE	MECHANICAL SPECIFICATION
2-1	1.1 Scope.....
2-1	1.2 General.....
2-3	1.3 Board Sizes and Dimensions.....
2-3	1.3.1 Printed Board Layout Considerations.....
2-3	1.3.2 Board Mounted Accessories.....
2-4	1.3.3 Board to Board Relationships.....
2-5	1.4 Connectors.....
2-7	1.5 Front Panels.....
2-11	1.6 Backplanes.....
	APPENDIX A
	RECOMMENDED DESIGN PRACTICES
2-11	A.1 General.....
2-11	A.2 Backplane Designs.....
2-11	A.2.1 11X" 11 Backplane Design.....
2-18	A.2.2 19" 19" Backplane Design.....
	TABLES
2-15	A-1 11X" 11 Backplane Characteristics.....
2-18	A-2 Connectors Necessary at 15 Connector.....
2-19	A-3 19" 19" Backplane Characteristics.....
	FIGURES
2-1	1-1 Board Sizes and System/Functional Assignment to Connectors.....
2-2	1-2 Typical Sub-Back.....
2-4	1-3 Double-High Board Sizes.....
2-5	1-4 Single-High Board Sizes.....
2-6	1-5 Ejector Tab.....
2-7	1-6 Board-to-Board Spacing Relationships.....
2-8	1-7 Hard Connector (603-2-18C-0096).....
2-8	1-8 Backplane Connector (603-2-18C-0096).....
2-9	1-9 Relationship of Mechanical Components.....
2-10	1-10 Double-High Front Panel.....
2-11	1-11 Single-High Front Panel.....
2-12	1-12 Backplane and Mounting Rail Relationship.....
2-13	1-13 Typical Backplane.....
2-16	A-1 Four-Slot 11X" 11 Backplane.....
2-17	A-2 Specifications for the 15 Connector.....
A-20	A-3 Twelve-Slot 19" Backplane.....
A-21	A-4 Twenty-Slot 19" Backplane.....



## CHAPTER 5 MECHANICAL SPECIFICATION

### 1.1 SCOPE

This section lists and defines the mechanical specifications that a designer must be concerned with to ensure that a Multibus II system is dimensionally compatible with International Electrotechnical Commission (IEC) standards. The dimensions and drawings within this specification are based on IEC standards. The four main parts of this section provide mechanical specifications for connectors, Multibus II boards, front panels, and backplanes. A discussion of recommended backplane design practices appears in Appendix A at the end of this section.

### 1.2 GENERAL

The Multibus II boards, board accessories, and backplanes must be designed and manufactured to the following specifications:

- IEC-603-2 - for Connectors and connector mounting
- IEC-297-3 - for Sub-rack and Board Design

If there are discrepancies between this specification and the IEC specifications, this specification should overrule.

Figure 1-1 shows the two boards defined for the Multibus II architecture, single-high and double-high, and shows the assignment of functions and busses to the connectors. Further explanations of the terms single-high and double-high appear later in the text.

The iPSB Bus and iLBX II Bus portions of the Multibus II bus architecture are always defined on the P1 and P2 connectors, respectively. However, the user can optionally define the use of the P2 connector. As depicted in Figure 1-1, the single-high board contains only the P1 connector; the double-high board contains both the P1 and the P2 connectors.

The example system of Figure 1-2 illustrates the physical relationships between sub-racks, backplanes, printed boards, and accessories. Boards reside vertically in the sub-rack with the component side facing the right and the iPSB connector above the iLBX II connector. The iPSB Backplane and the iLBX II Backplane are mounted onto the sub-rack. The double-high board is connected to both the iPSB Backplane and the iLBX II Backplane. The single-high board is mounted on top of a supporting divider and is connected to the iPSB Backplane.

V

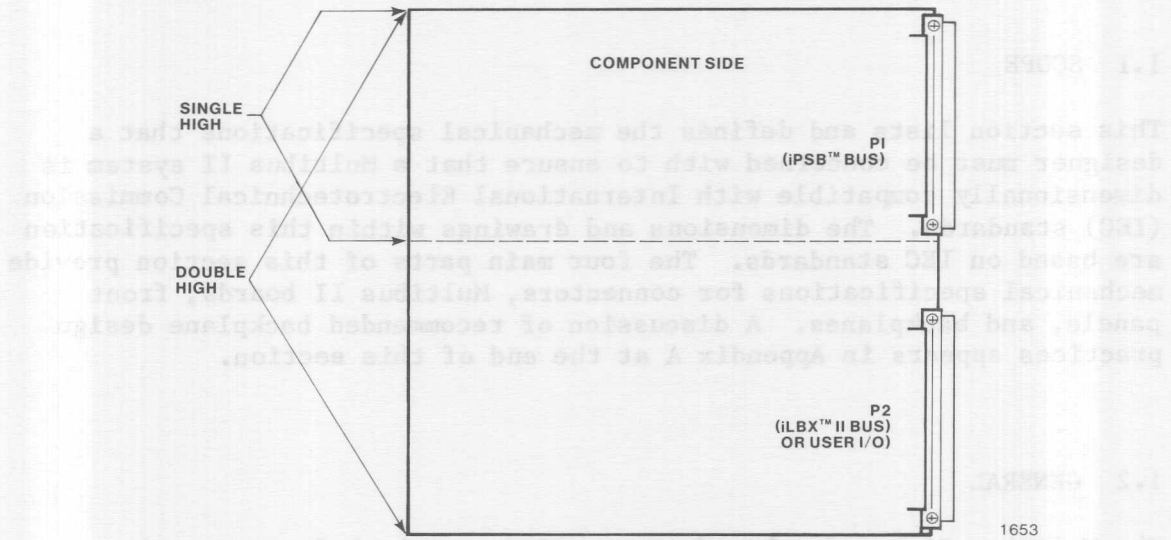


Figure 1-1. Board Sizes And Bus/Functional Assignment To Connectors

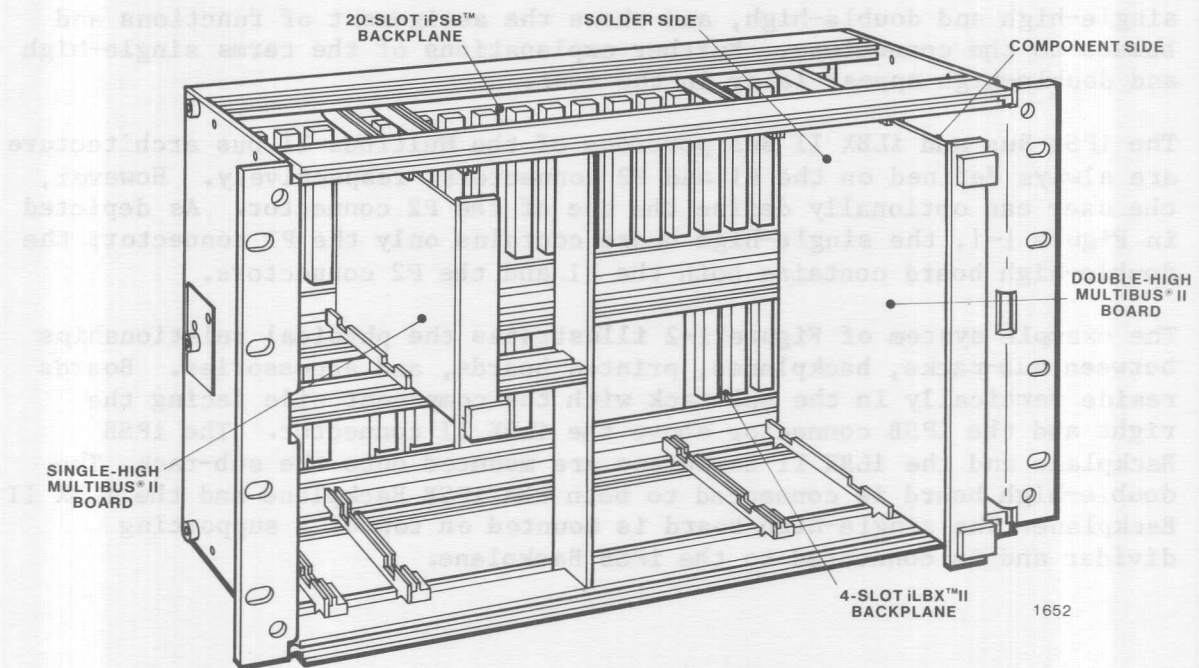


Figure 1-2. Typical Sub-Rack



### 1.3 BOARD SIZES AND DIMENSIONS

Multibus II boards are designed for compatibility with sub-racks designed according to IEC 297-3. Figures 1-3 and 1-4 show the double-high and single-high board form factors defined for a Multibus II system. Board depth, the distance from the front to the back of the board, is approximately 8.66" (220mm).

As defined in IEC 297-3, the double-high board height corresponds to the 6U board height (233.4 mm); and the single-high board height corresponds to the 3U board height (100.0 mm). For simplicity, this specification refers to them as double-high and single-high boards.

#### 1.3.1 Printed Board Layout Considerations

The minimum distance between board top/bottom edges and traces or components is 0.098" (2.5mm). Figures 1-3 and 1-4 illustrate this distance.

#### 1.3.2 Board-Mounted Accessories

Front panel mounting brackets are the board mounted accessories for the front panels. Ejector tabs may be used in place of front panels on the boards. Figures 1-3 and 1-4 show reserved areas onto which the ejector tabs or front panel mounting brackets are installed.

Mounting holes used to mount ejector tabs and front panels are 0.110" (2.8mm) in diameter. As shown in Figures 1-3 and 1-4, the hole location on the board is standardized to allow ejector tabs to be interchanged with front panels.

**1.3.2.1 EJECTOR TABS.** Ejector tabs are standardized to allow compatibility between sub-racks of different manufacturers and Multibus II boards. Ejector tabs must meet the dimensional specifications of Figure 1-5.

**1.3.2.2 FRONT PANEL MOUNTING BRACKETS.** The front panel mounting brackets provide the minimum required clearance of 0.1" (2.54mm) between the front edge of the board and the rear face of the front panel. This minimum clearance guarantees adequate connector engagement.

The handles on the front panel of a double-high board are located at the top and bottom rather than at the middle and bottom of the front panel. This allows more space on the front panel for input/output connectors. A middle front panel mounting bracket may be required on double high boards to insure mechanical integrity.



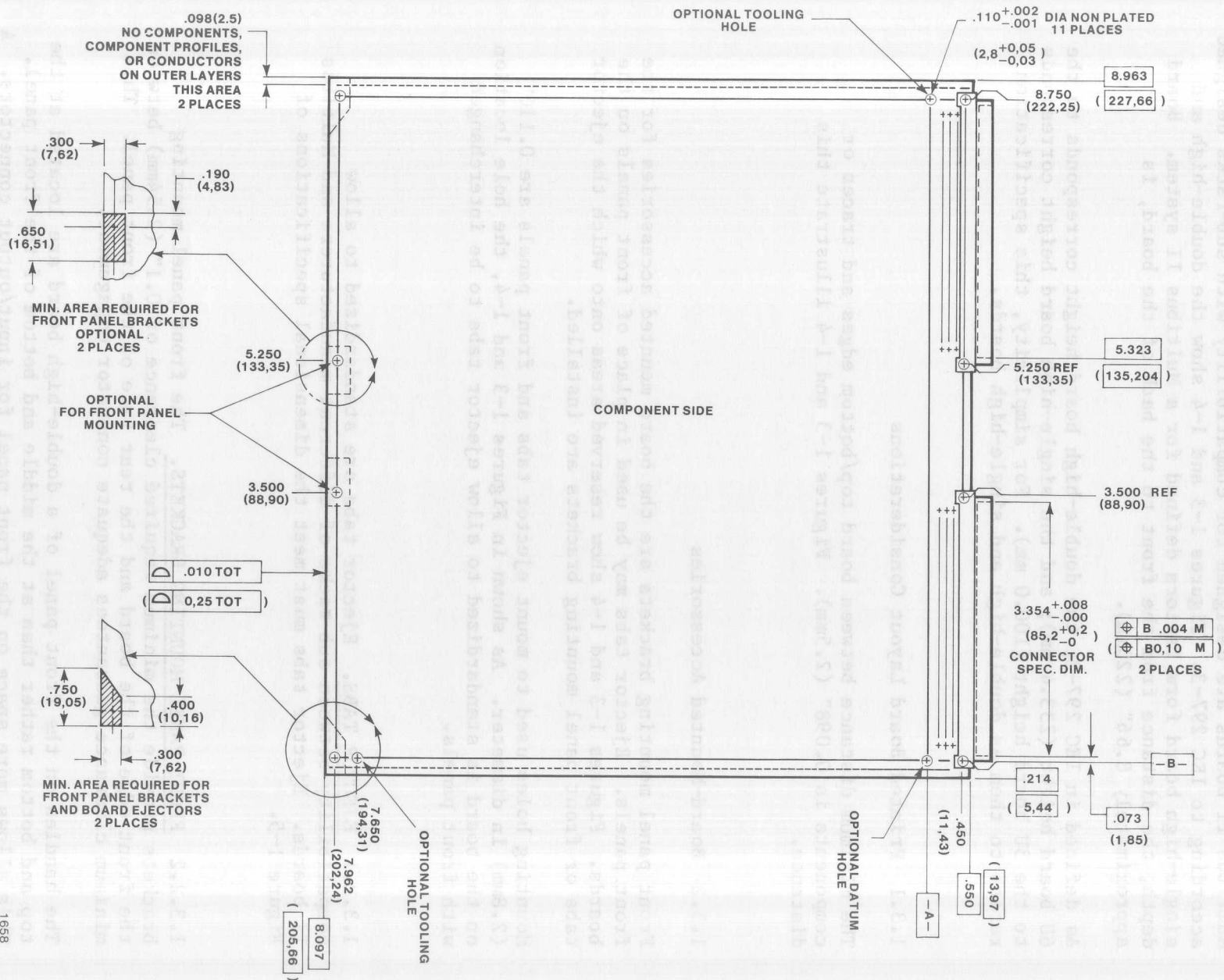


Figure 1-3. Double-High Board Size

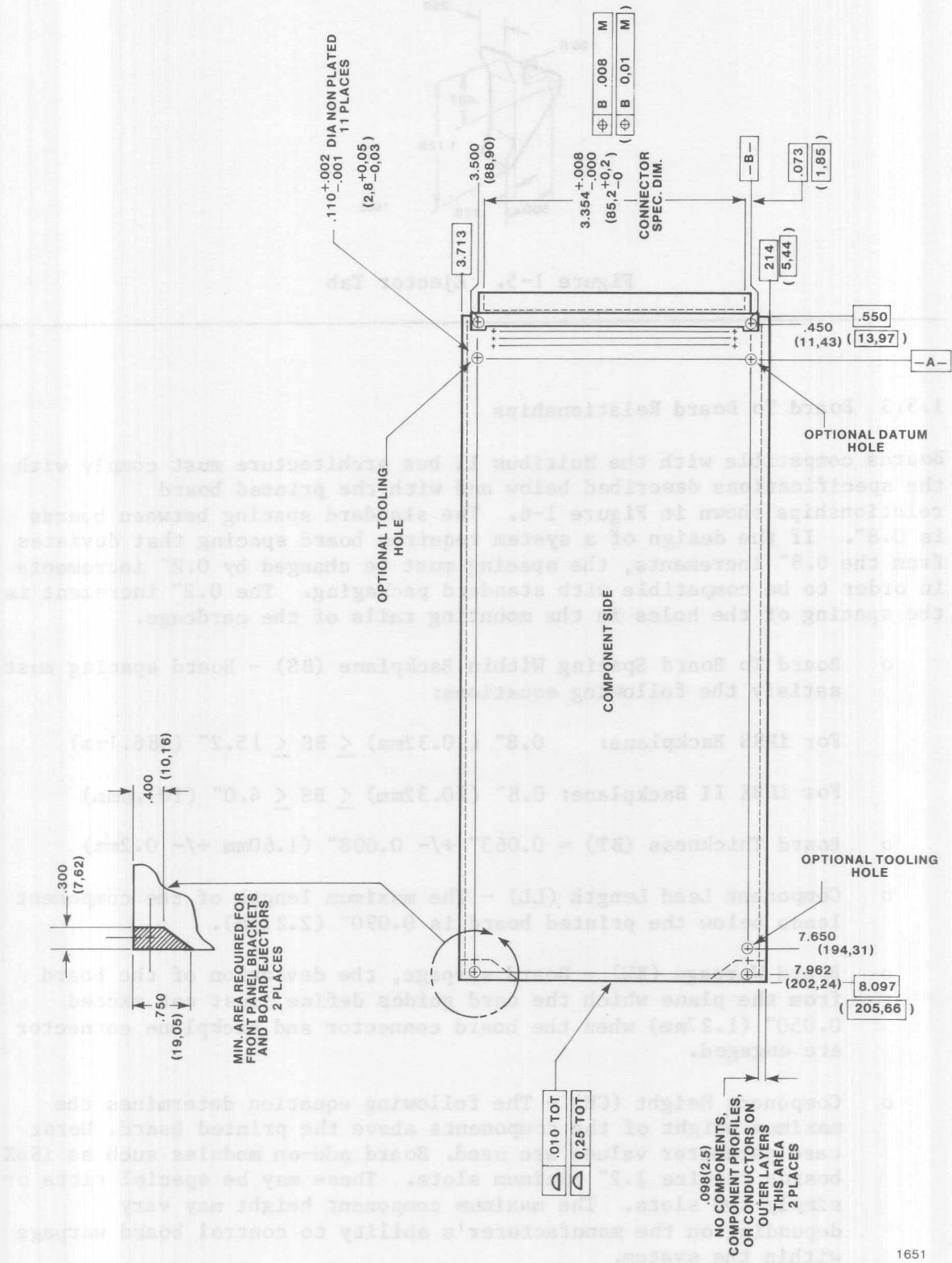


Figure 1-4. Single-High Board Size

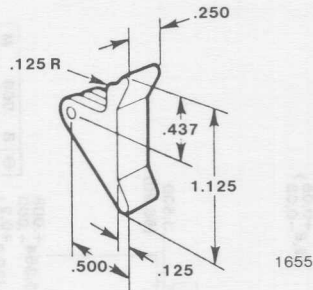


Figure 1-5. Ejector Tab

### 1.3.3 Board To Board Relationships

Boards compatible with the Multibus II bus architecture must comply with the specifications described below and with the printed board relationships shown in Figure 1-6. The standard spacing between boards is 0.8". If the design of a system requires board spacing that deviates from the 0.8" increments, the spacing must be changed by 0.2" increments in order to be compatible with standard packaging. The 0.2" increment is the spacing of the holes in the mounting rails of the cardcage.

- o Board To Board Spacing Within Backplane (BS) - Board spacing must satisfy the following equations:

For iPSB Backplane:  $0.8" (20.32\text{mm}) \leq BS \leq 15.2" (386.1\text{mm})$

For iLBX II Backplane:  $0.8" (20.32\text{mm}) \leq BS \leq 4.0" (101.6\text{mm})$

- o Board Thickness (BT) =  $0.063" \pm 0.008" (1.60\text{mm} \pm 0.2\text{mm})$
- o Component Lead Length (LL) - The maximum length of the component leads below the printed board is 0.090" (2.21mm).
- o Board Warpage (BW) - Board warpage, the deviation of the board from the plane which the card guides define, must not exceed 0.050" (1.27mm) when the board connector and backplane connector are engaged.
- o Component Height (CH) - The following equation determines the maximum height of the components above the printed board. Worst case parameter values are used. Board add-on modules such as iSBX boards require 1.2" minimum slots. These may be special slots or simply two slots. The maximum component height may vary depending on the manufacturer's ability to control board warpage within the system.

$$\begin{aligned} CH &< BS - (BT + LL + 2BW) \\ CH &< 0.80" - (0.071" + 0.090" + 0.10") \\ CH &< 0.542" (13.77\text{mm}) \end{aligned}$$

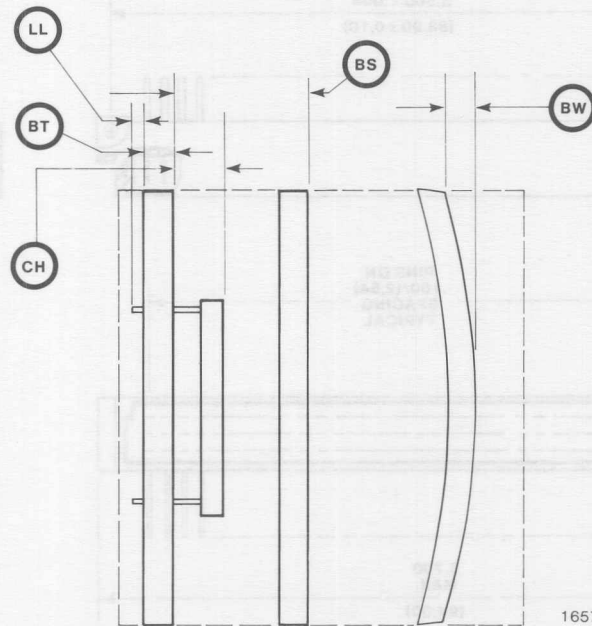


Figure 1-6. Board-To-Board Spacing Relationships

#### 1.4 CONNECTORS

Multibus II boards and backplanes use two-piece, 96-pin connectors for interconnection to the iPSB bus and the iLBX II bus. Figures 1-7 and 1-8 show the dimensional specifications of the connectors. The right-angle connectors on the printed board are IEC standard 603-2-IEC-C096-M; the receptacle connectors on the backplane are IEC standard 603-2-IEC-C096-F. Figure 1-9 illustrates the relationships between connectors and boards in the sub-rack.

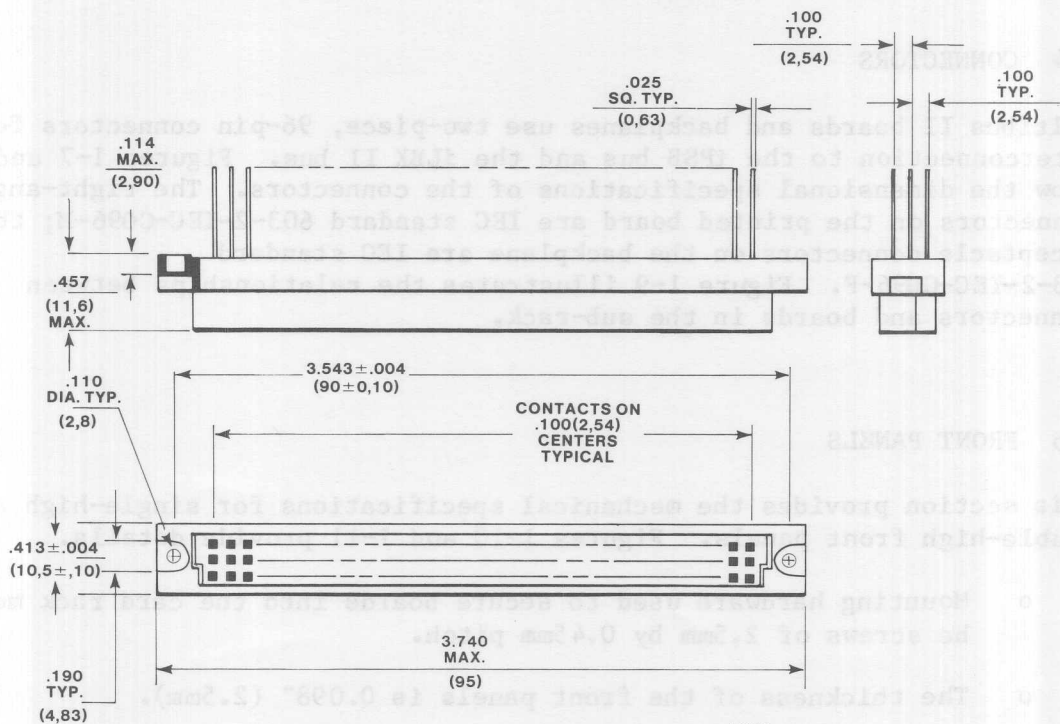
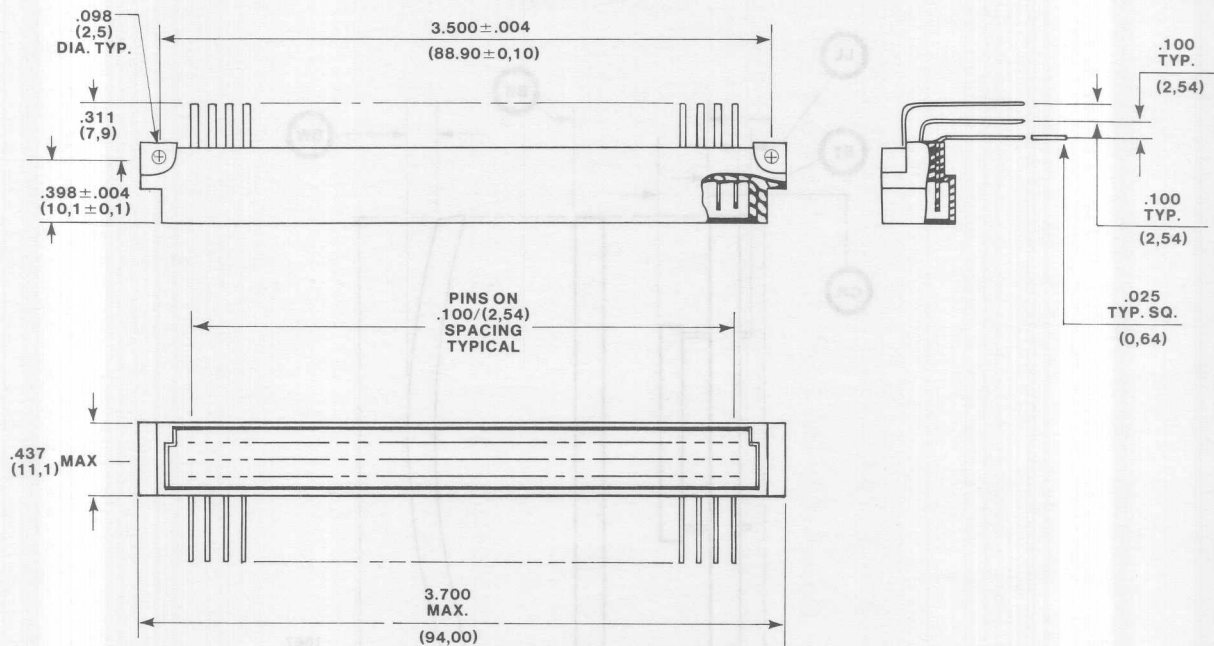
#### 1.5 FRONT PANELS

This section provides the mechanical specifications for single-high and double-high front panels. Figures 1-10 and 1-11 provide details.

- o Mounting hardware used to secure boards into the card rack must be screws of 2.5mm by 0.45mm pitch.
- o The thickness of the front panels is 0.098" (2.5mm).



# MECHANICAL SPECIFICATION



# MECHANICAL SPECIFICATION

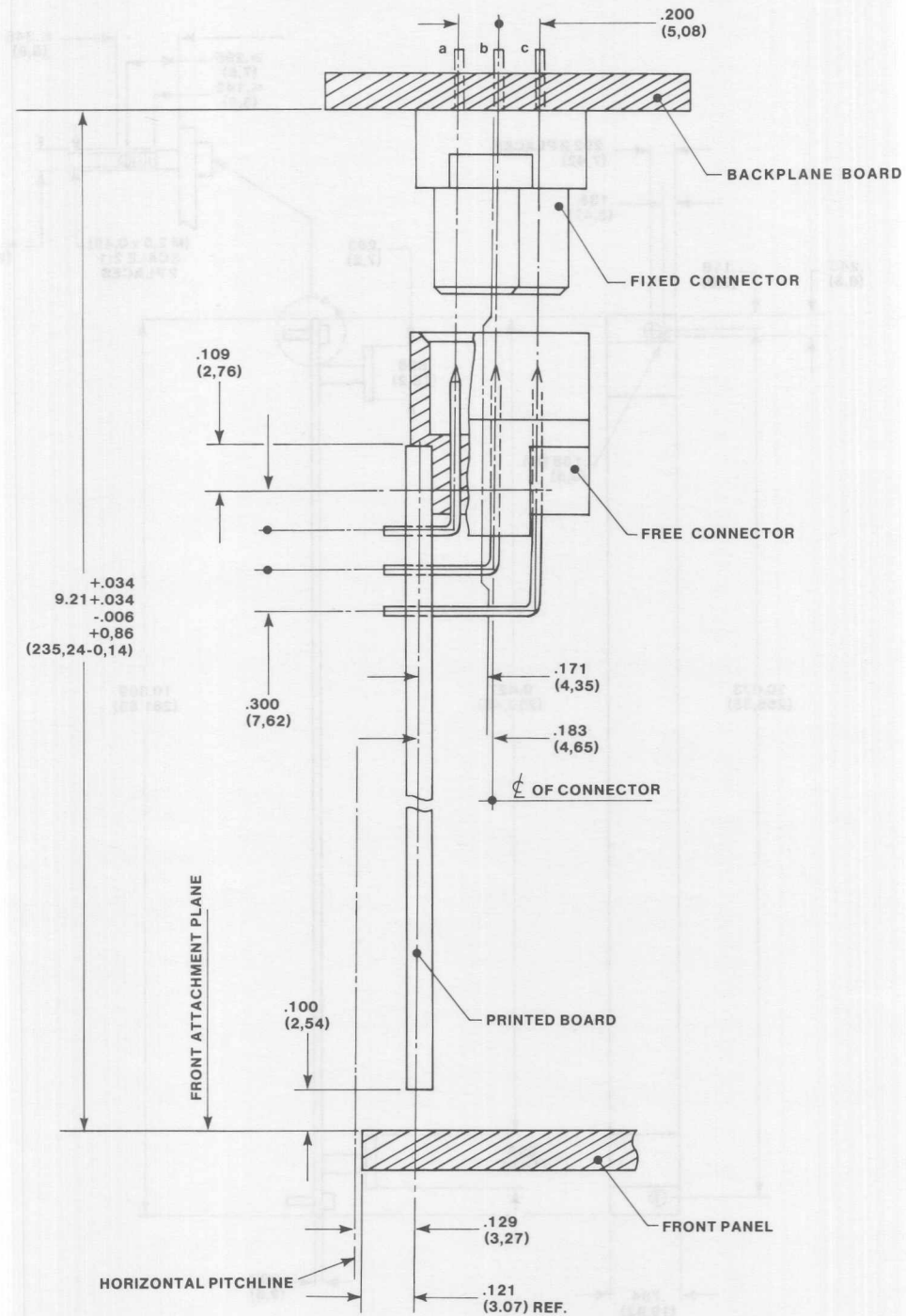


Figure 1-9. Relationship Of Mechanical Components

## MECHANICAL SPECIFICATION



Figure 1-10. Double-High Front Panel

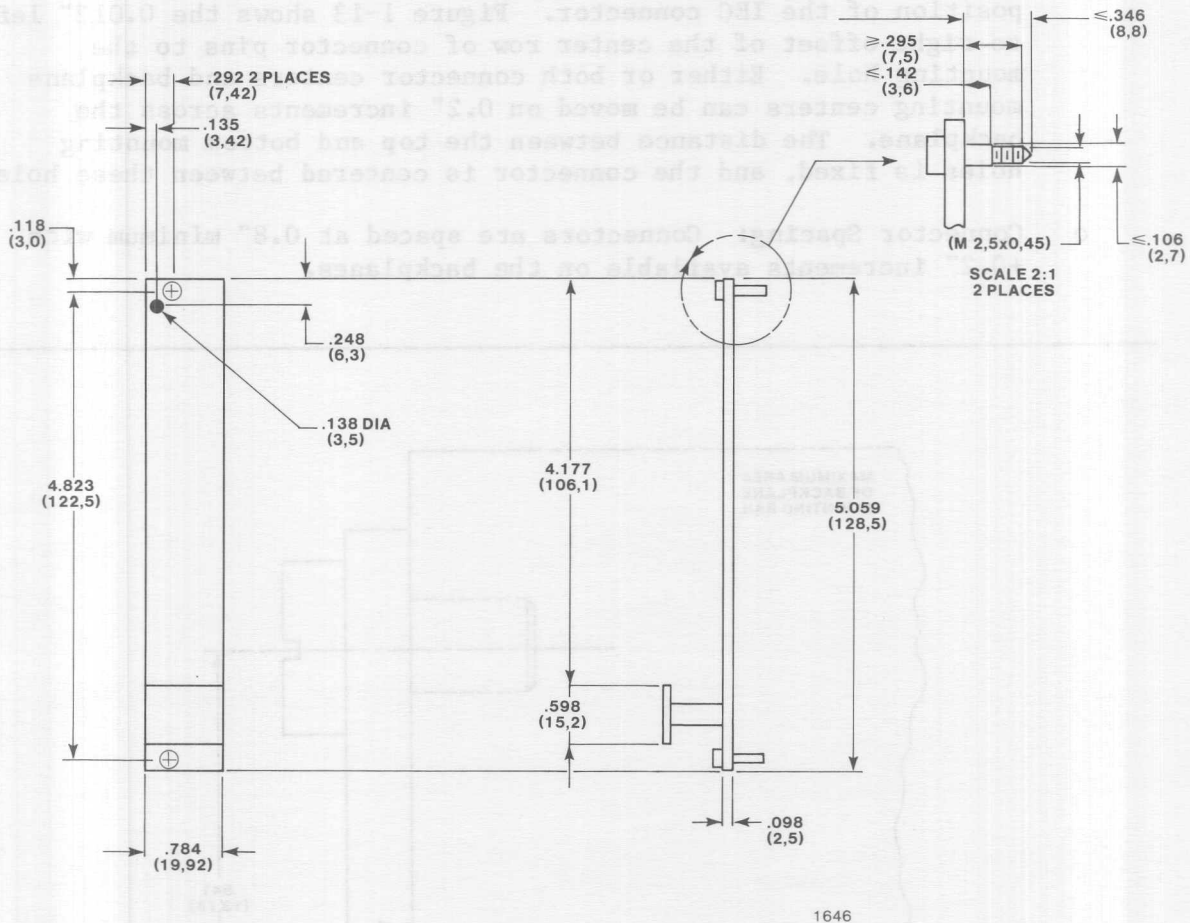


Figure 1-11. Single-High Front Panel

## 1.6 BACKPLANES

This section discusses the layout and dimensioning of Multibus II Backplanes. More specifically, the discussions cover the type of backplane fixing screws needed, the relationship between backplane and mounting rail positioning, and the relationship between mounting holes and connector positions.

- o Backplane Fixing screws: Backplanes are mounted to the sub-rack with standard 2.5mm by 0.45mm pitch screws.
- o Relationship of Backplane To Mounting Rails: Figure 1-12 shows the maximum area between the top of the 96-pin connector on the backplane and the mounting rail on the backplane. This dimension is important when designing power supply connections located between the top of the connector and the rail.



## MECHANICAL SPECIFICATION

- o Relationship of Mounting Holes to Connector Position: There is a fixed relationship between the backplane mounting hole and the position of the IEC connector. Figure 1-13 shows the 0.012" left to right offset of the center row of connector pins to the mounting hole. Either or both connector centers and backplane mounting centers can be moved on 0.2" increments across the backplane. The distance between the top and bottom mounting holes is fixed, and the connector is centered between these holes.
- o Connector Spacing: Connectors are spaced at 0.8" minimum with +0.2" increments available on the backplanes.

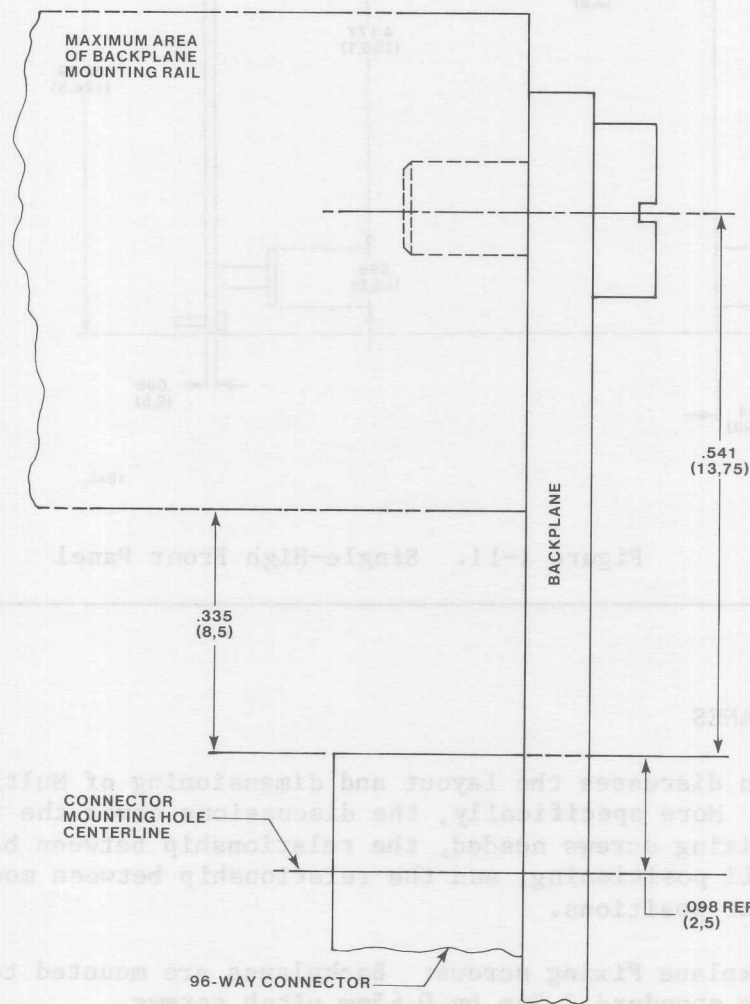


Figure 1-12. Backplane And Mounting Rail Relationship

# MECHANICAL SPECIFICATION

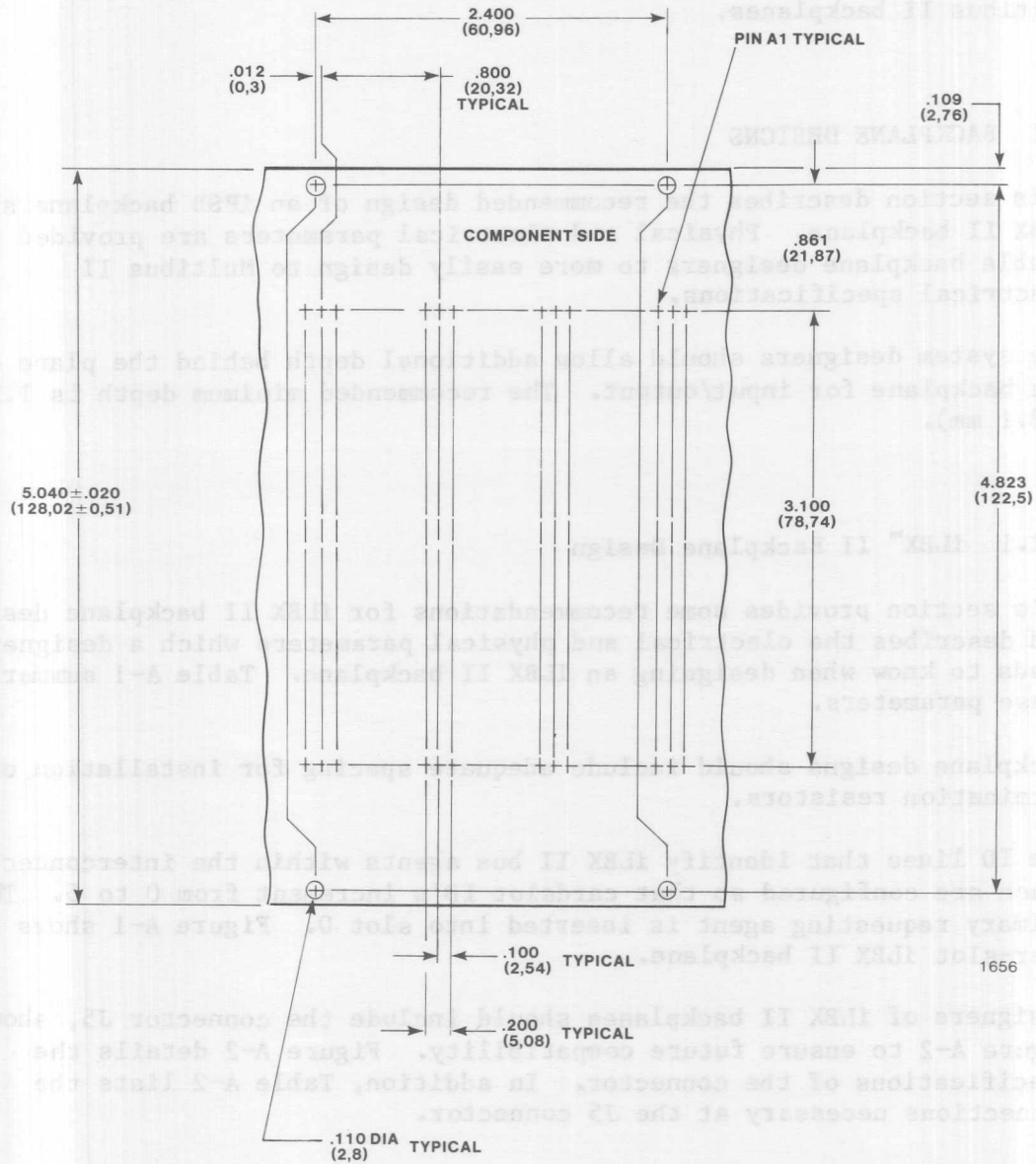


Figure 1-13. Typical Backplane

## APPENDIX A. RECOMMENDED DESIGN PRACTICES

## A.1 GENERAL

This appendix provides details of recommended design approaches for Multibus II backplanes.

## A.2 BACKPLANE DESIGNS

This section describes the recommended design of an iPSB backplane and an iLBX II backplane. Physical and electrical parameters are provided to enable backplane designers to more easily design to Multibus II electrical specifications.

The system designers should allow additional depth behind the plane of the backplane for input/output. The recommended minimum depth is 1.50" (38.1 mm).

## A.2.1 iLBX™ II Backplane Design

This section provides some recommendations for iLBX II backplane design and describes the electrical and physical parameters which a designer needs to know when designing an iLBX II backplane. Table A-1 summarizes these parameters.

Backplane designs should include adequate spacing for installation of termination resistors.

The ID lines that identify iLBX II bus agents within the interconnect space are configured so that cardslot ID's increment from 0 to 5. The primary requesting agent is inserted into slot 0. Figure A-1 shows the four-slot iLBX II backplane.

Designers of iLBX II backplanes should include the connector J5, shown in Figure A-2 to ensure future compatibility. Figure A-2 details the specifications of the connector. In addition, Table A-2 lists the connections necessary at the J5 connector.

Table A-1. iLBX™ II Backplane Characteristics

Characteristics	Parameter
Trace-to-trace Spacing	100 mils
Trace Width	0.01 inch
Number of Planes	6
Unloaded Impedance	60 to 85 ohms
Loaded Impedance	15 to 20 ohms
Termination	Refer to LBX II Bus Specification
Copper Thickness	2 oz.
Backplane Thickness	0.125 inch

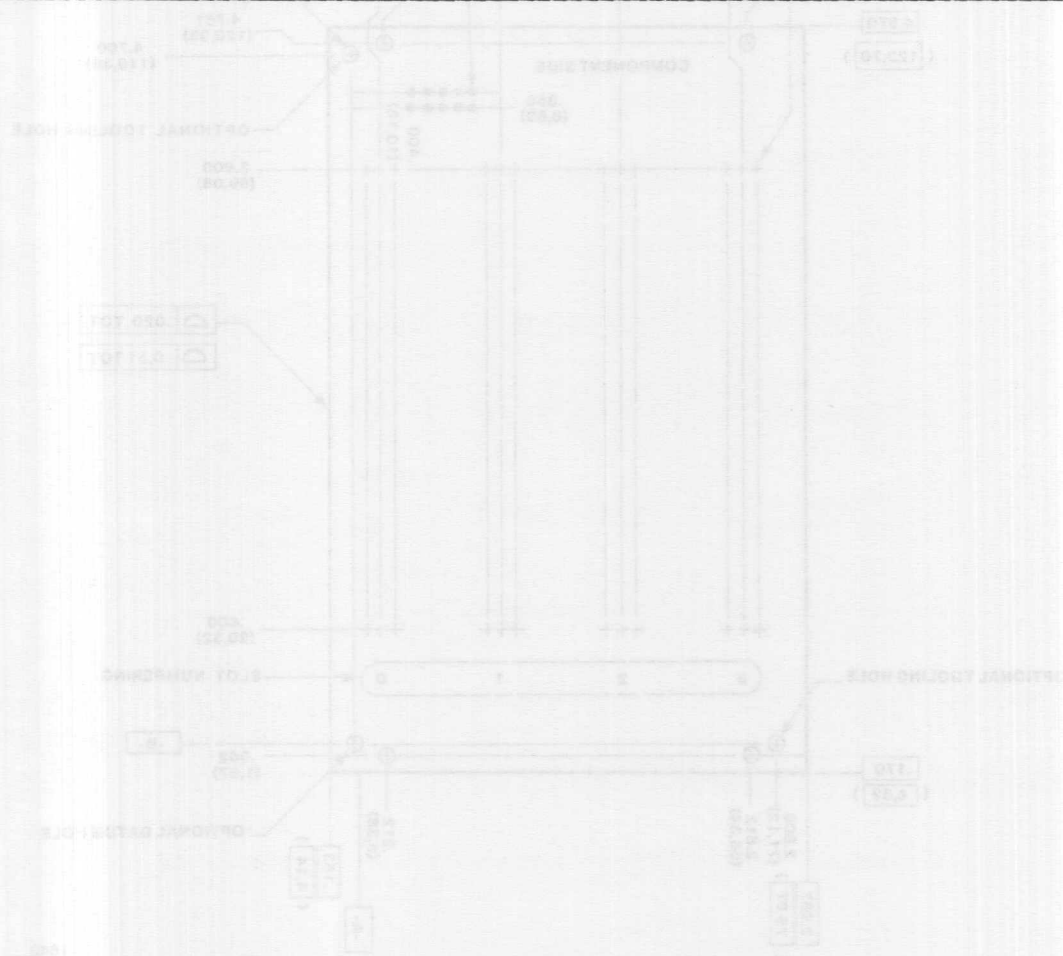


Figure A-1. Four-Side iLBX™ II Backplane



# MECHANICAL SPECIFICATION

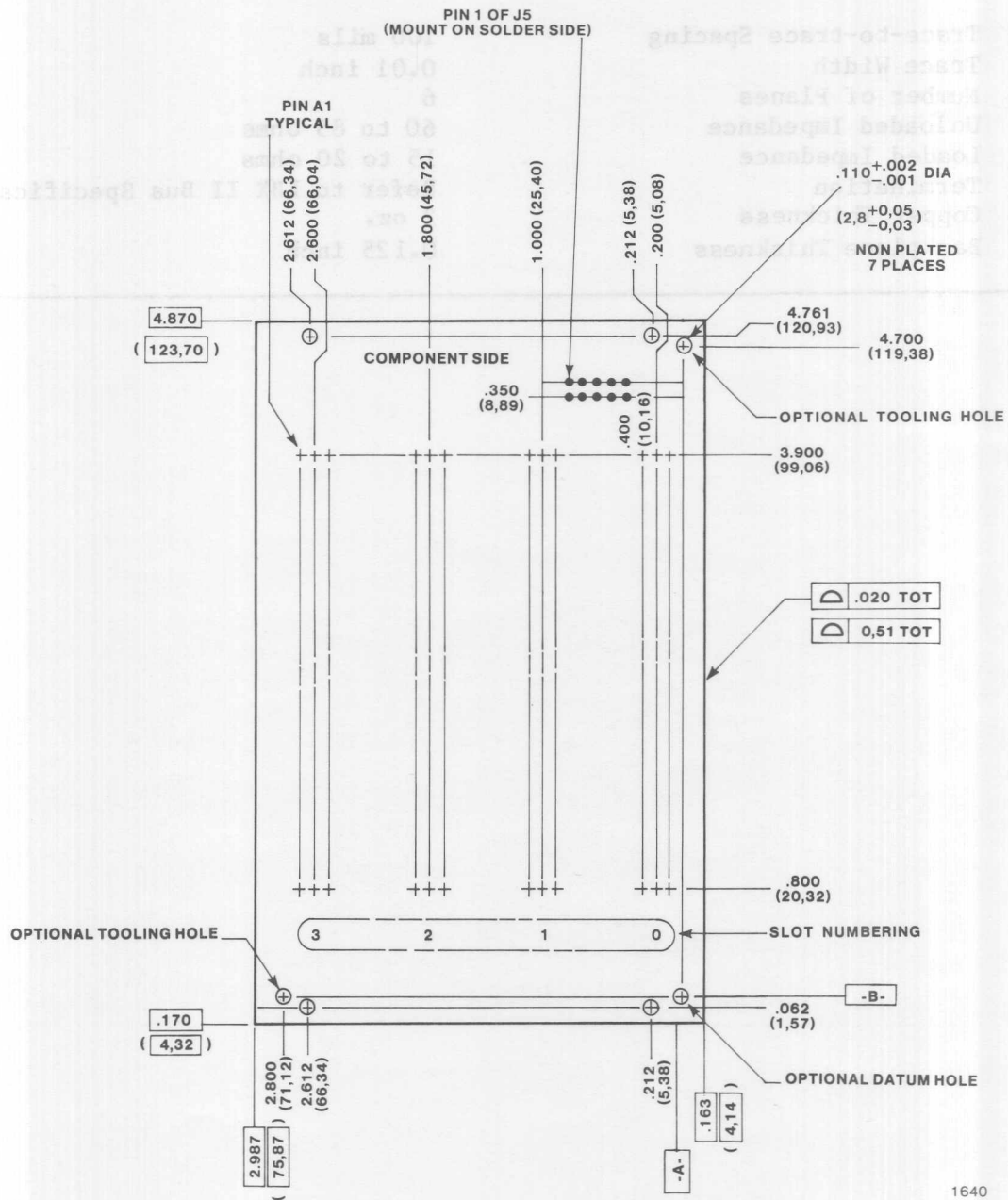


Figure A-1. Four-Slot iLBX™ II Backplane

# MECHANICAL SPECIFICATION

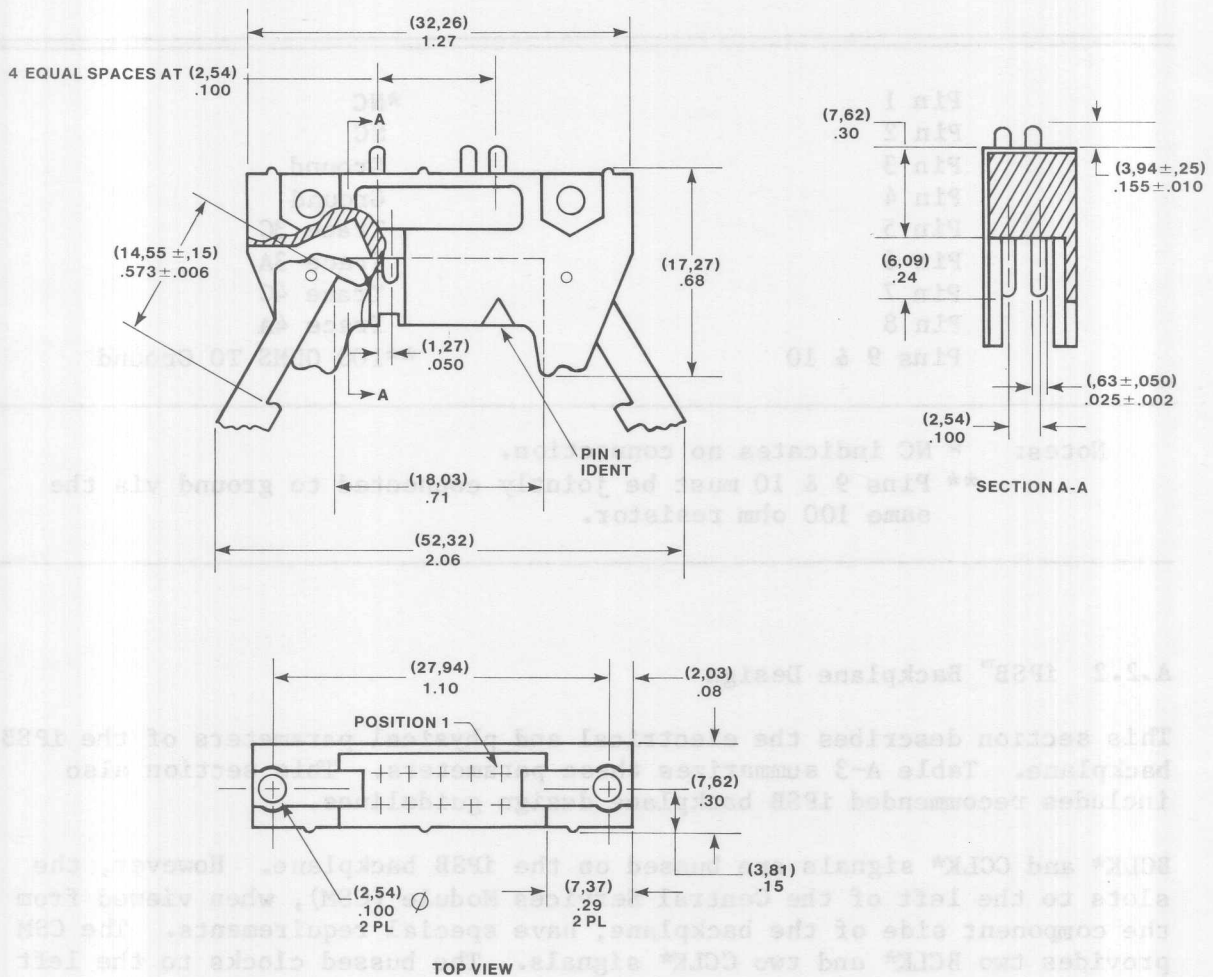


Figure A-2. Specifications For The J5 Connector

# MECHANICAL SPECIFICATION

Table A-2. Connections Necessary At J5 Connector

Pin Number	Connection
Pin 1	*NC
Pin 2	NC
Pin 3	Ground
Pin 4	Ground
Pin 5	Trace 3C
Pin 6	Trace 3A
Pin 7	Trace 4C
Pin 8	Trace 4A
Pins 9 & 10	**100 OHMS TO Ground

Notes: \* NC indicates no connection.

\*\* Pins 9 & 10 must be jointly connected to ground via the same 100 ohm resistor.

## A.2.2 iPSB™ Backplane Design

This section describes the electrical and physical parameters of the iPSB backplane. Table A-3 summarizes these parameters. This section also includes recommended iPSB backplane design guidelines.

BCLK\* and CCLK\* signals are bussed on the iPSB backplane. However, the slots to the left of the Central Services Module (CSM), when viewed from the component side of the backplane, have special requirements. The CSM provides two BCLK\* and two CCLK\* signals. The bussed clocks to the left of the CSM connect to the clocks driven on the A row of the iPSB connector containing the CSM. The bussed clock signals to the right of the CSM connect to the clocks driven on the C row of the iPSB connector containing the CSM.

Figures A-3 and A-4 show the slot numbering sequence. The CSM always occupies slot 0. Beginning at the CSM and proceeding to the right, the cardslots are numbered in sequence. From the right end of the cardcage, the numbering sequence is "wrapped around" to continue at the far end until reaching the CSM cardslot.

The 20 slot backplane shown in Figure A-4 fits within a standard 19 inch sub-rack. Termination is included at each end of the backplane.

# MECHANICAL SPECIFICATION

Table A-3. iPSB™ Backplane Characteristics

Characteristic	Parameter
Trace-to-trace Spacing	100 mils
Trace Width	0.01 inch
Number of Planes	6
Unloaded Impedance	60 to 85 ohms
Loaded Impedance	15 to 20 ohms
Termination	Refer to iPSB Bus Specification
Copper Thickness	2 oz.
Backplane Thickness	0.125 inch

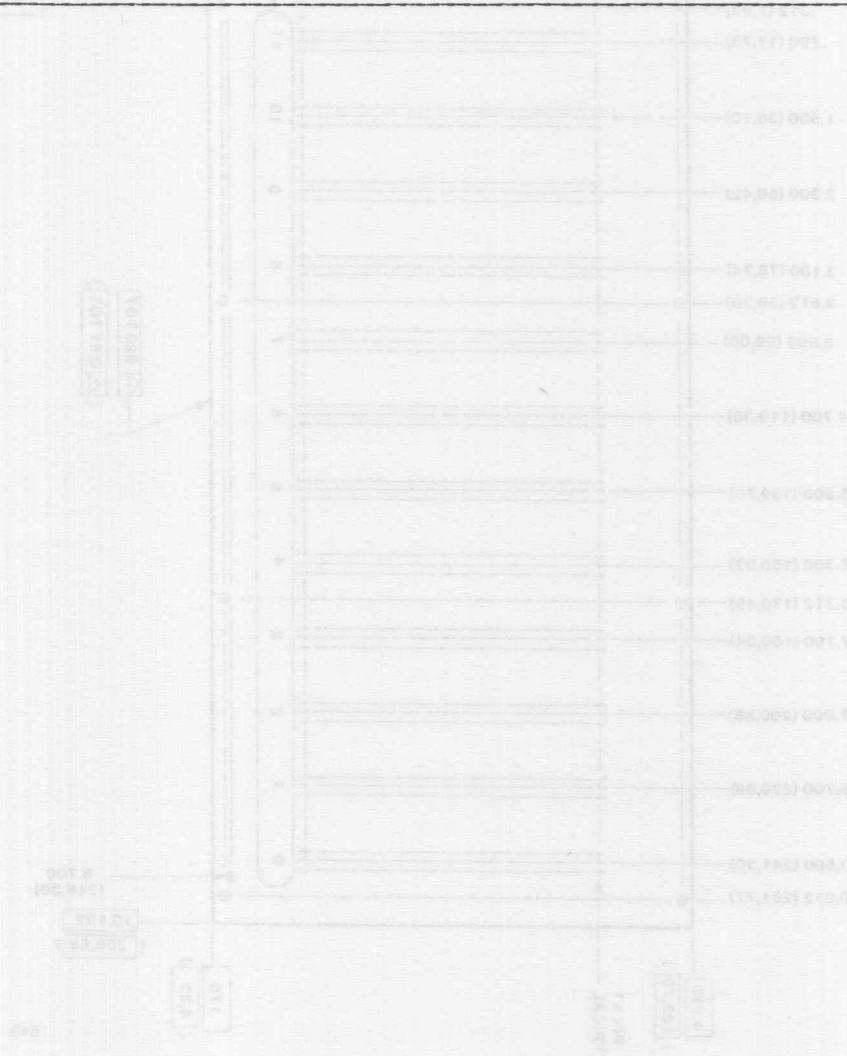


Figure A-3. Twelve-Slot iPSB™ Backplane



# MECHANICAL SPECIFICATION

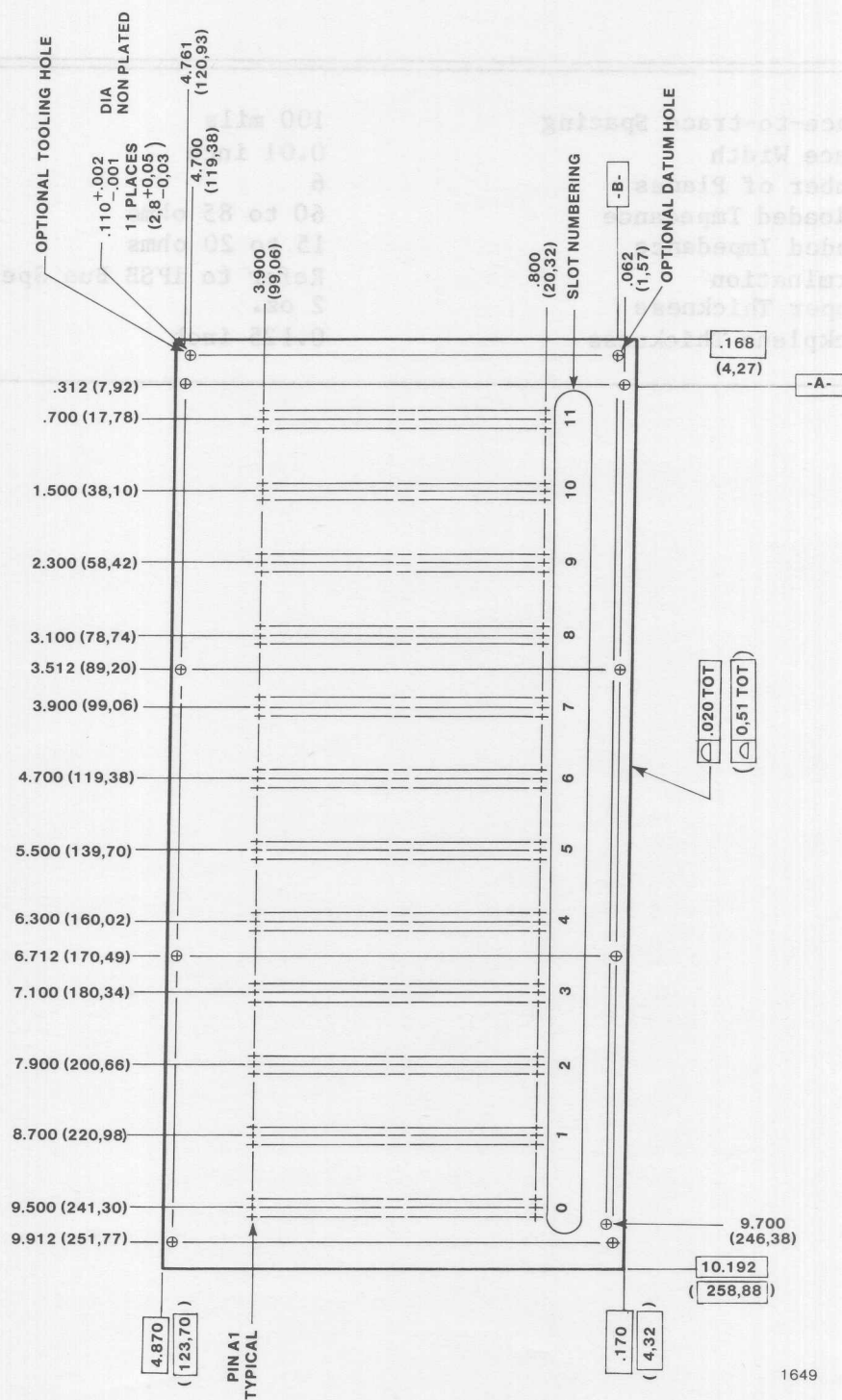


Figure A-3. Twelve-Slot iPSB™ Backplane

# MECHANICAL SPECIFICATION

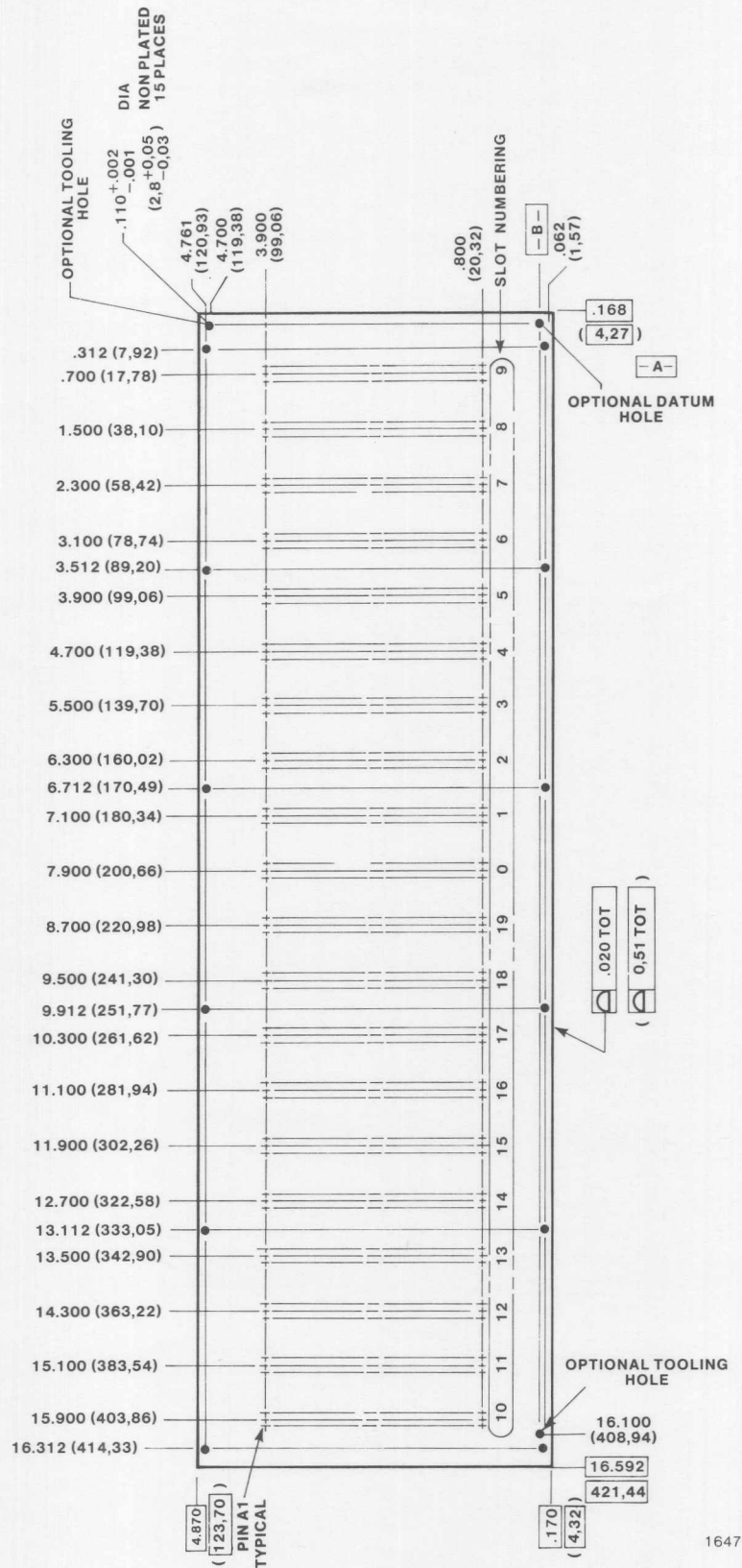


Figure A-4. Twenty-Slot iPSB™ Backplane

\*\*\*





# **INTEL MULTIBUS® II SYSTEM INTERFACE SPECIFICATION**

---







# CONTENTS

	PAGE
SYSTEM INTERFACE SPECIFICATION	
1.1 Scope.....	6-1
1.2 Object.....	6-1
1.3 Architecture Overview.....	6-1
1.4 Definitions.....	6-2
1.5 The Interface Model.....	6-3
1.5.1 Initiator Module.....	6-5
1.5.2 Selector Module.....	6-5
1.5.3 Multiplexor Module.....	6-5
1.5.4 Receiver Module.....	6-5
1.6 Address Space Descriptions.....	6-6
2. Interconnect Space Operation.....	6-7
2.1 Introduction.....	6-7
2.1.1 Interconnect Address Alignment.....	6-8
2.1.2 Vendor Identification Registers.....	6-8
2.1.3 User-Defined Registers.....	6-9
2.1.4 Example of Using the Interconnect Address.....	6-9
2.2 Interface Model for Interconnect Space.....	6-11
2.2.1 View from the Requesting Agent.....	6-11
2.2.2 View from the Replying Agent.....	6-12
3. I/O Address Space Operations.....	6-13
3.1 Introduction.....	6-13
3.2 Interface Model for I/O Address Space.....	6-14
3.2.1 View from the Requesting Agent.....	6-14
3.4.2 View from the Replying Agent.....	6-15
4. Memory Address Space Operations.....	6-16
4.1 Introduction.....	6-16
4.2 Interface Model for Memory Address Space.....	6-17
4.2.1 View From The Requesting Agent.....	6-17
4.2.2 View From The Replying Agent.....	6-18
5. Message Space Operations.....	6-19
5.1 Introduction.....	6-19
5.2 Interface Model.....	6-20
5.2.1 Local Bus Interface.....	6-21
5.2.2 iSSB™/iPSB™ Interface.....	6-21
5.3 Message Model.....	6-23
5.3.1 Unsolicited Messages.....	6-23
5.3.2 Solicited Messages.....	6-25



## CONTENTS (continued)

	PAGE
SYSTEM INTERFACE SPECIFICATION	

5.4 Unsolicited Message Transfer.....	6-27
5.5 Solicited Message Transfer.....	6-27
5.5.1 Negotiation Phase.....	6-28
5.5.2 Transfer Phase.....	6-29
5.5.3 Completion Phase.....	6-29
5.6 Packet Formats.....	6-31

### TABLES

1-1. Address Spaces Available For Agents In A MULTIBUS® II System	6-6
2-1. Interconnect Address Space Summary.....	6-7
3-1. I/O Address Space Summary.....	6-13
4-1. Memory Address Space Summary.....	6-16
5-1. Message Address Space Summary.....	6-19
5-1. Packet Formats On The Bus.....	6-31

### FIGURES

1-1. MULTIBUS® II Bus Architecture.....	6-2
1-2. MULTIBUS® II System Interface Model.....	6-4
2-1. Interconnect Space Format.....	6-8
2-2. Interconnect Template For Memory Board Example .....	6-10
2-3. Interconnect Register Contents For Memory Board Example....	6-10
2-4. Interconnect Space Interface Model.....	6-11
3-1. I/O Space Interface Model.....	6-14
4-1. Memory Address Space Interface Model.....	6-17
5-1. Message Space Interface Model.....	6-20
5-2. Typical iPSB™ Packet.....	6-22
5-3. Typical iSSB™ Packet.....	6-23
5-4. Typical Unsolicited Transfer.....	6-27
5-5. Typical Solicited Message Transfer.....	6-30
5-6. Unsolicited Output Message Packet.....	6-32
5-7. Unsolicited Output Broadcast Message Packet.....	6-33
5-8. Reset Message Packet.....	6-34
5-9. Buffer Request Message Packet.....	6-35
5-10. Buffer Grant Message Packet.....	6-36
5-11. Data Packet.....	6-37



## CHAPTER 6

# SYSTEM INTERFACE SPECIFICATION

### 1.1 SCOPE

The Multibus II System Interface specification provides information on how the various bus structures work together to provide a system architecture.

This specification presents the system architecture considerations required for building a Multibus II system that uses more than one of the bus structures within the architecture. The purpose of the System Interface Specification is to ensure direct connect of two separate implementations of a Multibus II system or direct connection of a future system implementation with a current one.

The specification applies to Multibus II microprocessor computer systems or portions of them where:

- o An implementation of the Multibus II system requires more than one of the interfaces defined in the Multibus II bus architecture.
- o The implementation of a current Multibus II system may be redesigned or upgraded in the future, but must operate compatibly with the current system implementation.

### 1.2 OBJECT

The Multibus II bus architecture defines four separate address spaces within a system. Those address spaces are called Interconnect, I/O, Memory, and Message. Not all address spaces are supported on each bus. This System Interface Specification describes the address spaces and identifies the busses on which the an agent can access the address space.

### 1.3 ARCHITECTURE OVERVIEW

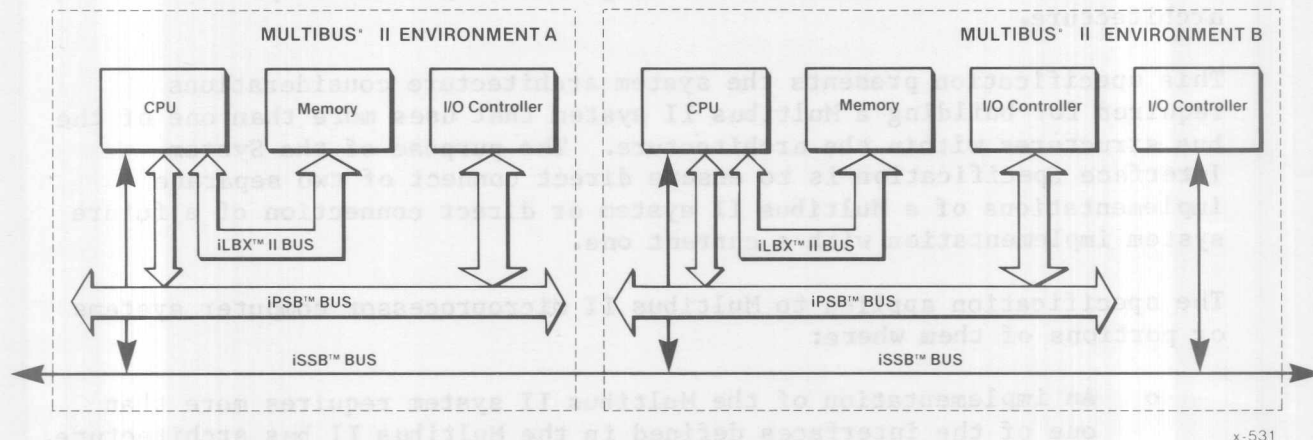
The Multibus II bus architecture provides five interfaces that are used to construct a system. Those interfaces are:

- o the Parallel System Bus (iPSB bus) interface
- o the Local Bus Extension Bus (iLBX II bus) interface
- o the Serial System Bus (iSSB bus) interface
- o the Single Board Extension Bus (iSBX bus) interface, and
- o the Multichannel bus interface



## SYSTEM INTERFACE SPECIFICATION

In many cases, a task can be performed satisfactorily on more than one interface. Figure 1-1 shows how the interfaces work together in a Multibus II bus architecture.



x-531

Figure 1-1. MULTIBUS® II Bus Architecture

This chapter of the Multibus II specification defines the uses of the three new busses within the Multibus II bus architecture: the Parallel System bus (iPSB bus), the Local Bus Extension (iLBX II bus), and the Serial System bus (iSSB bus). In defining uses for each bus, the text develops a model to describe the requirements for transferring information on the different interfaces within the architecture.

### 1.4 DEFINITIONS

The following definitions apply to the System Interface Specification. This section contains only general definitions; more specific definitions are provide in other sections as appropriate.

**Requesting Agent:** The agent that initiates a transfer operation on one of the busses. The requesting agent is comprised of two modules, the Initiator module and the Selector module.

**Initiator Module:** The module within a requesting agent that initiates an operation on the media.

## SYSTEM INTERFACE SPECIFICATION

Selector Module:	The module within the requesting agent that sends the operation onto the media and directs it to the proper destination.
Replying Agent:	The agent that cooperates with the requesting agent to complete the transfer operation. On any transfer operation, at least one replying agent is involved. Certain busses in the Multibus II architecture place restrictions on replying agents and their involvement with transfer operations. The replying agent is composed of two modules, the Multiplexor module and the Receiver module.
Multiplexor Module:	The module within replying agents that decodes the destination address and decides whether or not to receive an operation from the bus.
Receiver Module:	The module within a replying agent that performs the operations that its local Multiplexor module accepts from the bus.

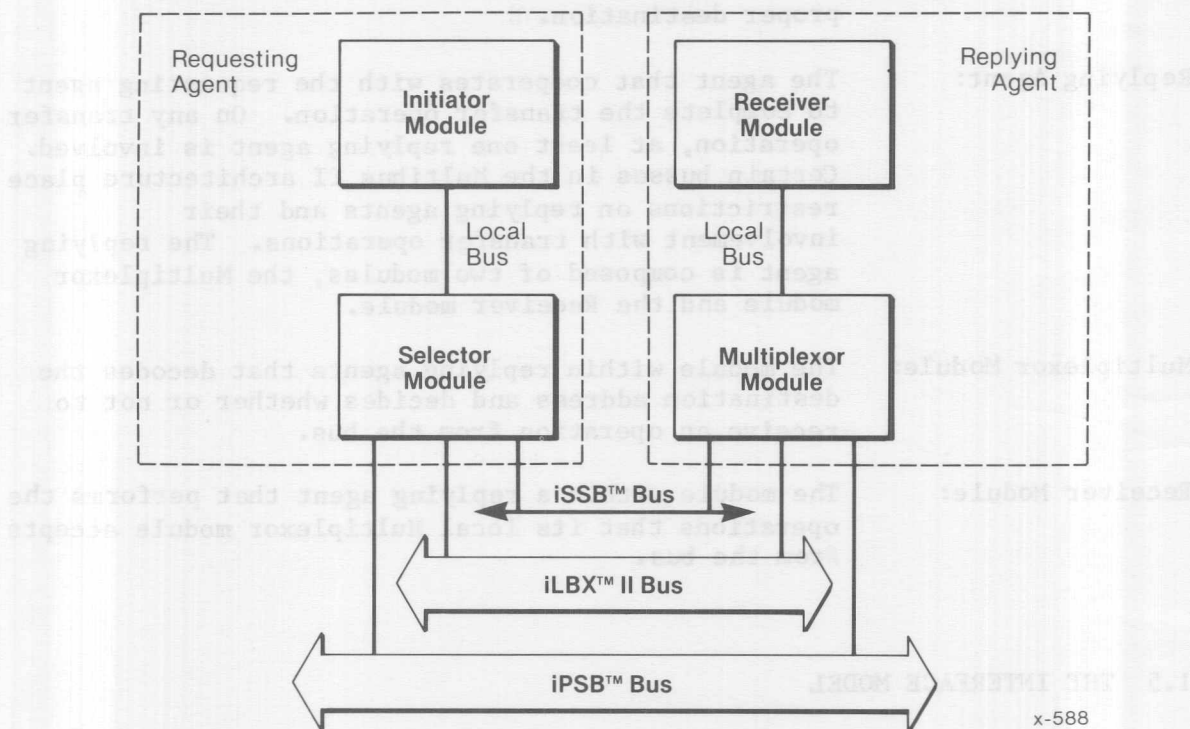
### 1.5 THE INTERFACE MODEL

Figure 1-2 shows the interface model for the Multibus II bus architecture, including the iPSB bus, iLBX II bus and iSSB bus. The figure shows both the requesting agent and the replying agent.

The requesting agent consists of two modules, the Initiator module and the Selector module. The Initiator module initiates an operation by assembling an operation. Then the Initiator module places the operation into a buffer within the Selector module. The Selector module examines the operation and determines which of the three busses will carry the operation. In doing so, the Selector module isolates the Initiator module from the particulars of the bus protocols. This presents the Initiator module with a uniform software interface for all operations.

The replying agent also consists of two modules: the Multiplexor module and the Receiver module. The Multiplexor module monitors each bus interface on the agent. The functions performed by the Multiplexor module depend on which bus or busses connect to the agent. The Multiplexor module receives operations from a bus and buffers them for the Receiver module. The Receiver module performs the operation requested by the Initiator module.

## SYSTEM INTERFACE SPECIFICATION



x-588

Figure 1-2. MULTIBUS® II System Interface Model

In several cases, agents have a choice as to which transfer media to use in performing an operation. As an example, a message space operation may be routed either serially on the iSSB bus or in parallel on the iPSB bus. Each type of operation in the Multibus II bus architecture must satisfy three requirements regardless of the bus through which it is routed. The requirements are:

- o Each Initiator module must be able to unambiguously select a bus over which to route an operation. The determination is based solely on local addressing within the agent.
- o The Selector module must recognize the bus selection done by the Initiator module without direct intervention by the Initiator module.
- o The protocol required on each bus must be invisible to the software in the Initiator and Receiver modules.

The hardware circuitry within the Selector module and the Multiplexor module is significantly reduced by including in a Multibus II system only those busses that are required. For most operations, the Multibus II bus architecture allows transparent interchange of busses as system requirements change; examples are message space operation interchanges from the iPSB bus to the iSSB bus and memory space operation interchanges from the iPSB bus to the iLBX II bus.

## SYSTEM INTERFACE SPECIFICATION

### 1.5.1 Initiator Module

The Initiator module on the requesting agent is the initiator of a data transfer operation. It is responsible for supplying to the Selector module all information necessary for the Receiver module to perform the transfer operation. The Initiator module passes media-independent information to the Selector module for transmission. The content of this information identifies the transfer media for the operation.

### 1.5.2 Selector Module

The Selector module on the requesting agent identifies the transfer media via the information from the Initiator module. The bus selection by the Selector module may be either a static or a dynamic selection based on the information from the Initiator module. The Selector module separates the four address spaces and routes operations at each address space onto the proper bus.

### 1.5.3 Multiplexor Module

The Multiplexor module on the replying agent monitors the activity of the busses to which the agent is connected. The module is responsible for following all bus protocols and for actively monitor operations on the busses. However, only those operations that address the Multiplexor module are received and buffered.

If addresses, the Multiplexor module stores the operation; if not, the Multiplexor module ignores the operation. The Multiplexor module may translate data from a media-dependent form on the bus to a media-independent form for use by the Receiver module. Some implementations of replying agents may require that the Multiplexor module perform operations independent of the Receiver module and on receiving a command from the Selector module (such as a reset operation).

### 1.5.4 Receiver Module

The Receiver module within the replying agent performs the operation in most cases. More than one replying agent may be involved in a single transfer operation; therefore, more than one Receiver module may be active at any given instant. After it receives an operation, the Multiplexor module notifies the Receiver module that the operation is present. The Receiver module obtains the operation from the buffer and performs it; specifics of the operation depend on the purpose of the Receiver module.



# SYSTEM INTERFACE SPECIFICATION

## 1.6 ADDRESS SPACE DESCRIPTIONS

The Multibus II bus architecture supports four address spaces:

- o Interconnect Used for board identification, system configuration, and board specific functions such as testing, diagnostics, and configuration.
- o Memory Used for accessing physical memory devices for data and code storage and retrieval.
- o I/O Used for accessing peripheral devices such as communication controllers and mass storage devices.
- o Message Used for inter-module, inter-agent, and interprocessor communications ranging from interrupts to negotiated data movement.

Each of the address spaces is defined on at least one of the three busses in the Multibus II system. Table 1-1 shows the configurations that are allowed among the four address spaces and the five busses within the Multibus II bus architecture. In a case where several busses share access to one address space, the system designer assigns each module a unique address (as seen by the initiator module) through which it is accessed.

In addition to the three busses in the Multibus II bus architecture, the iSBX bus is accessed through the I/O space and the Multichannel bus is accessed through a combination of the I/O space and the memory space. The iSBX bus and the Multichannel bus are carry-over architectures from the original Multibus architecture. As such, both the iSBX bus and the Multichannel bus interfaces are defined in separate specifications and not covered within this document.

Table 1-1. Address Spaces Available For Agents In A MULTIBUS® II System

Agent on the	Interconnect Space	Memory Space	I/O Space	Message Space
iPSB Bus	Yes	Yes	Yes	Yes
iLBX II Bus	Yes	Yes	No	No
iSSB Bus	No	No	No	Yes
iSBX Bus	No	No	Yes	No
Multichannel Bus	No	Yes	Yes	No

# SYSTEM INTERFACE SPECIFICATION

## 2. INTERCONNECT SPACE OPERATION

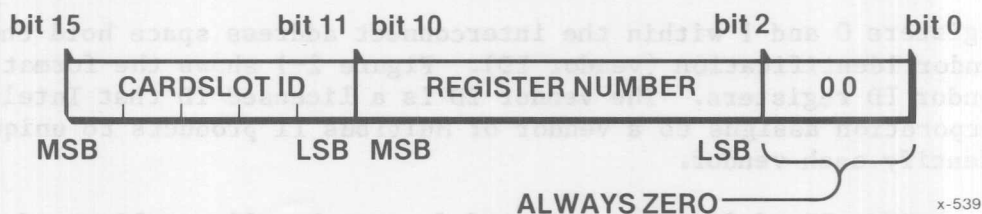
### 2.1 INTRODUCTION

The purpose of interconnect space in the Multibus II bus architecture is to allow flexible system configuration and diagnostic capability. Agents use the interconnect space to initialize, configure, control, test, and monitor board-specific functions. Agents on the iPSB bus or the iLBX II bus require a dedicated interconnect address. In a system, the interconnect space for an iPSB bus interface on an agent may (but is not required to) be the same physical address space as the interconnect space for an iLBX II bus interface on that same agent.

Table 2-1 gives an overview of the attributes available for an agent operating in interconnect address space. The description includes a figure showing the format for the address that is required within the interconnect address space.

Table 2-1. Interconnect Address Space Summary

- o Supports agents on both the iLBX II bus and the iPSB bus.
- o Supports 8-bit data transfers.
- o Supports both read and write operations to the interconnect space.
- o Does not support data transfer operations other than selection of one of 512 registers within the agent.
- o Supports point-to-point operations (sequential access and broadcast operations are not supported).
- o Supports cardslot ID of 0 to 19 for iPSB bus.
- o Supports cardslot ID of 24 to 29 for iLBX II bus.
- o Supports 16-bit addresses (5-bits for cardslot ID address, 9-bits for register number, and 2-bits that are always zero), as follows:



## SYSTEM INTERFACE SPECIFICATION

### 2.1.1 Interconnect Address Assignment

Each agent that has an iPSB bus and/or an iLBX II bus interface must have an interconnect address. On both the iLBX II and the iPSB bus, the interconnect ID is the cardslot ID that is assigned on power-up.

Each cardslot in the iPSB or iLBX II backplane is assigned an interconnect address. The iLBX II cardslot IDs are 24 through 29. The iLBX II cardslot ID of 24 always contains the primary requesting agent. The iPSB bus cardslot IDs are 00 through 19. The iPSB cardslot ID of 00 always contains the agents with the CSM functions.

The interconnect address consists of two parts: a cardslot ID and a sequence of up to 512 register numbers at that cardslot. Support of registers 0 and 1 (the Vendor ID Registers) within each cardslot is required if the cardslot contains a bus agent. Support of registers 2 through 511 (user-defined) is optional.

Figure 2-1 shows a diagram of the register set within the interconnect address for an agent. Each agent in a Multibus II system has a separate interconnect address and a separate set of interconnect registers.

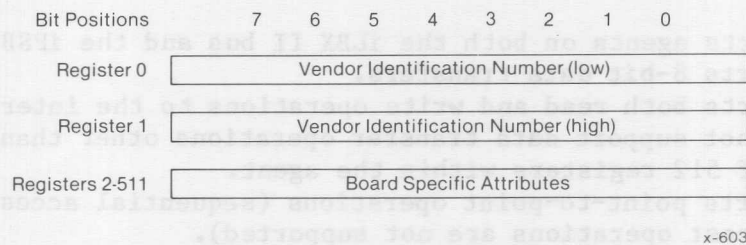


Figure 2-1. Interconnect Space Format

### 2.1.2 Vendor Identification Registers

Registers 0 and 1 within the interconnect address space hold the 16-bit vendor identification (vendor ID). Figure 2-1 shows the format of the vendor ID registers. The vendor ID is a licensed ID that Intel Corporation assigns to a vendor of Multibus II products to uniquely identify each vendor.

The vendor ID of 0000H is reserved for use by all non-licensed vendors.

Interconnect registers 0 and 1 are read-only registers; write operations to these registers are ignored.

## SYSTEM INTERFACE SPECIFICATION

### 2.1.3 User-Defined Registers

Registers 2 through 511 are user-defined within the interconnect space. These registers may be read/write, write-only (e.g., a reset register), or read-only registers, depending on the application requirements. This allows more flexible configuration of the board.

### 2.1.4 Example Of Using The Interconnect Address

Figures 2-2 and 2-3 provide an example of an agent using Registers 0 through 8 in the interconnect space. The agent in the example is a memory board with an interconnect register set defined as shown. The quantity and functions of the interconnect registers are defined by the vendor of the memory board.

In this example, registers 0 and 1 contain the vendor ID, 4321H. Following the vendor ID, registers 2 and 3 contain the board ID, 1234H. Registers 4 and 5 contain the revision number for the board, 05H. The next two registers (registers 6 and 7) define the starting 64k byte address boundary for the on-board memory space and the last register (register 8) defines the quantity of 64K byte blocks of memory on-board.

Of these registers, only the starting memory address registers (Registers 6 and 7) and the memory size register (Register 8) are read/write registers. All others are read-only registers. All registers should be initialized on power-up and Registers 6, 7, and 8 may be reconfigured during normal board operation if required.

Figure 2-2. Interconnect Register Set for Memory Board Example

Register	0	1	2	3	4	5	6	7	8
Register 0	0	0	0	0	0	0	0	0	0
Register 1	0	0	0	0	0	0	0	0	0
Register 2	0	0	0	0	0	0	0	0	0
Register 3	0	0	0	0	0	0	0	0	0
Register 4	0	0	0	0	0	0	0	0	0
Register 5	0	0	0	0	0	0	0	0	0
Register 6	0	0	0	0	0	0	0	0	0
Register 7	0	0	0	0	0	0	0	0	0
Register 8	0	0	0	0	0	0	0	0	0

Figure 2-3. Interconnect Register Contents for Memory Board Example



# SYSTEM INTERFACE SPECIFICATION

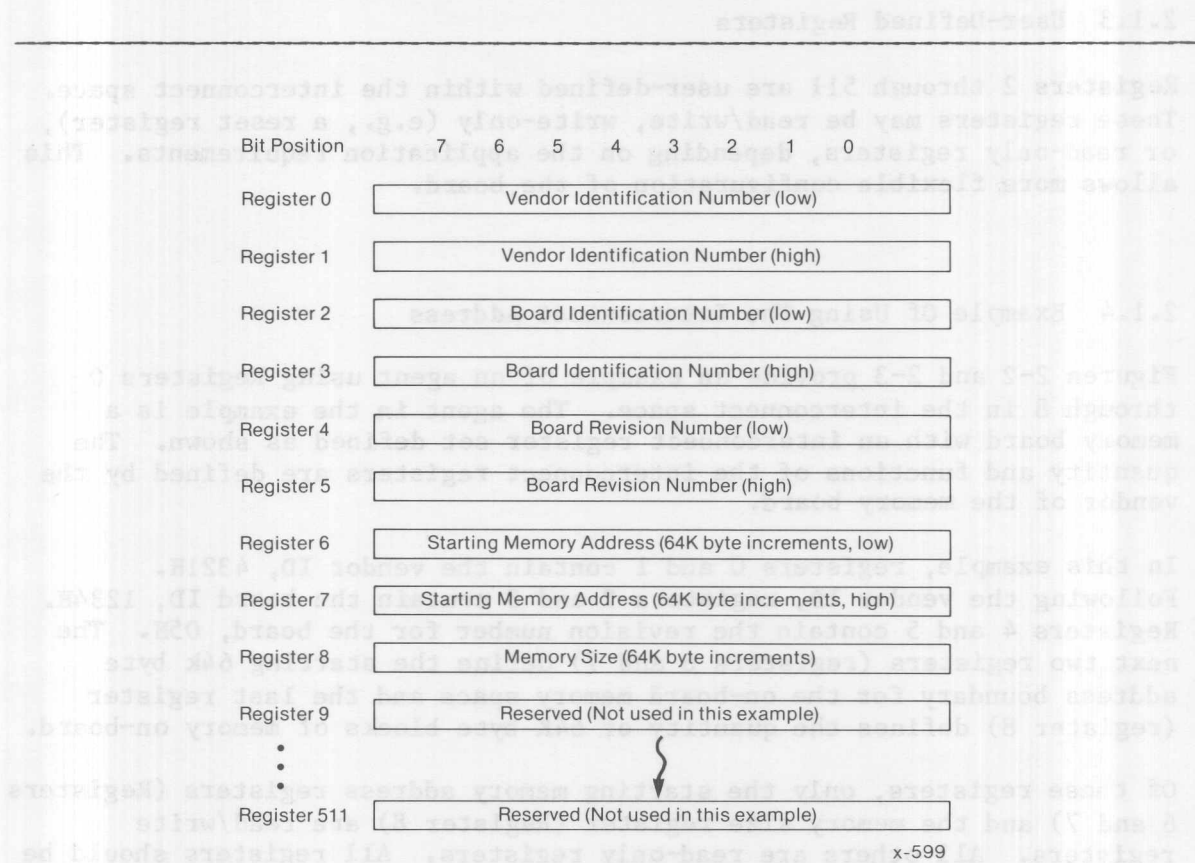


Figure 2-2. Interconnect Template For Memory Board Example

Bit Position	7	6	5	4	3	2	1	0
Register 0	0	0	1	0	0	0	0	1
Register 1	0	1	0	0	0	0	1	1
Register 2	0	0	1	1	0	1	0	0
Register 3	0	0	0	1	0	0	1	0
Register 4	0	0	0	0	0	1	0	1
Register 5	0	0	0	0	0	0	0	0
Register 6	Starting Memory Address (low order) Read/Write							
Register 7	Starting Memory Address (high order) Read/Write							
Register 8	Memory Size Read/Write							

x-605

Figure 2-3. Interconnect Register Contents For Memory Board Example

## SYSTEM INTERFACE SPECIFICATION

### 2.2 INTERFACE MODEL FOR INTERCONNECT SPACE

Figure 2-4 shows the model for an agent-to-agent operation in the interconnect space.

The interconnect space in a Multibus II system is partitioned by cardslot in the backplane. Within each cardslot ID, an agent may further partition the interconnect space into a maximum of 512 8-bit registers.

The interconnect space does not support sequential operations. This simplifies the logic within the Multiplexor module for interconnect space operations.

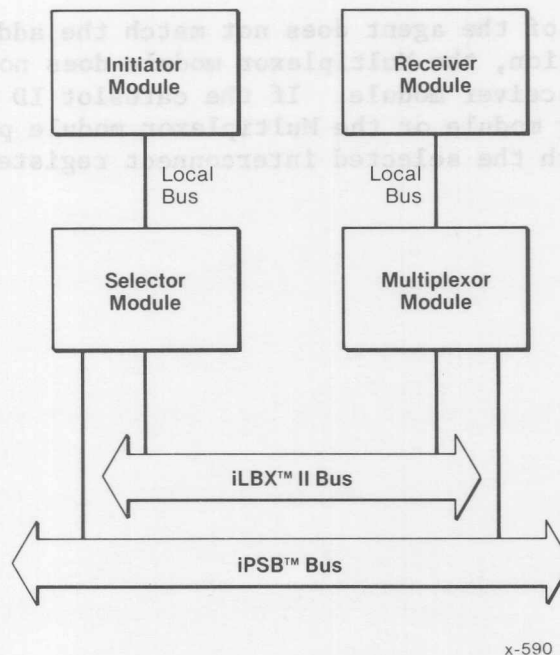


Figure 2-4. Interconnect Space Interface Model

#### 2.2.1 View From The Requesting Agent

On detecting an interconnect space operation, the Selector module converts the operation to a format compatible with the bus onto which the operation is targeted. The requesting agent performs an operation to a specific device within the interconnect space by referencing the interconnect address of that device. The interconnect address is simply a combination of the cardslot ID for the replying agent and a register number within the interconnect space for that replying agent.

## SYSTEM INTERFACE SPECIFICATION

An initiator module that does not directly access interconnect space may still use the interconnect space by referencing the interconnect addresses through a window, either in the processors memory space or I/O space. This method is referred to as either memory-mapping or I/O-mapping the interconnect space.

### 2.2.2 View From The Replying Agent

The Multiplexor module on the replying agent must be able to identify an interconnect operation that is directed to its Receiver module. The Multiplexor module does so by comparing its cardslot ID with that contained in the interconnect operation (for an iPSB bus operation). For an iLBX II bus operation, the Multiplexor module compares the cardslot ID from the interconnect operation with a dedicated value in the agent.

If the cardslot ID of the agent does not match the address in the interconnect operation, the Multiplexor module does not pass the operation to the Receiver module. If the cardslot ID does match, then either the Receiver module or the Multiplexor module performs the read or write operation with the selected interconnect register.

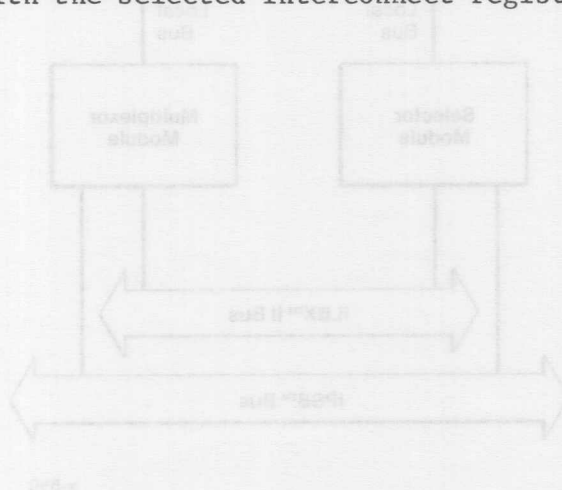


Figure 3-4. Interconnect Space Interface Model

### 3.1. View From The Requesting Agent

On detecting an interconnect space operation, the Initiator module converts the operation to a format compatible with the bus onto which the operation is targeted. The requesting agent performs an operation to a specific device within the interconnect space by referencing the interconnect address of that device. The interconnect address is simply a combination of the cardslot ID for the replying agent and a register number within the interconnect space for that replying agent.

## SYSTEM INTERFACE SPECIFICATION

### 3. I/O ADDRESS SPACE OPERATIONS

#### 3.1 INTRODUCTION

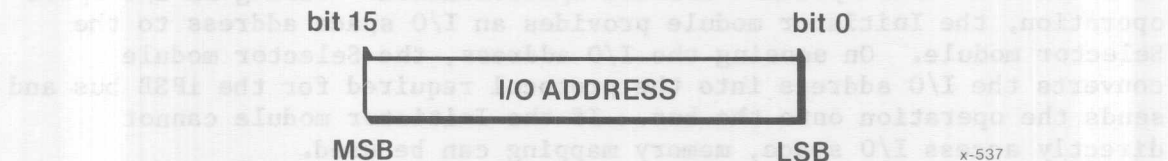
Agents use the I/O space in the Multibus II bus architecture for controlling peripheral devices on the iPSB bus. The typical I/O space operations pass commands and parameters to communications controllers or mass storage devices. I/O space operations never involve more than one replying agent.

In the Multibus II system, an I/O device has a unique address at which it is accessed. However, only iPSB bus agents can access I/O space; the iLBX II bus and the iSSB bus agents are not allowed access to I/O space in the Multibus II bus architecture.

Table 3-1 gives an overview of the attributes available for an agent operating in I/O address space. The description includes a figure showing the format for the address that is required within the I/O address space.

Table 3-1. I/O Address Space Summary

- o Supports agents on the iPSB bus only.
- o Supports both read and write operations.
- o Supports 8-bit, 16-bit, 24-bit, and 32-bit data transfer operations.
- o Supports sequential transfers to an I/O address with no address increment.
- o Supports point-to-point operations (broadcast operations are not supported).
- o Supports 16-bit addresses for an operation, as follows:





## SYSTEM INTERFACE SPECIFICATION

### 3.2 INTERFACE MODEL FOR I/O ADDRESS SPACE

Figure 3-1 shows the interface model for I/O space accesses in a Multibus II system. The I/O space allows different agents to use data widths of 8-bits, 16-bits, 24-bits, or 32-bits, and to perform either single or sequential accesses. In each case, the Multiplexor module identifies the type of operation. A typical function of an I/O space operation is to control a peripheral device such as serial communications or mass storage.

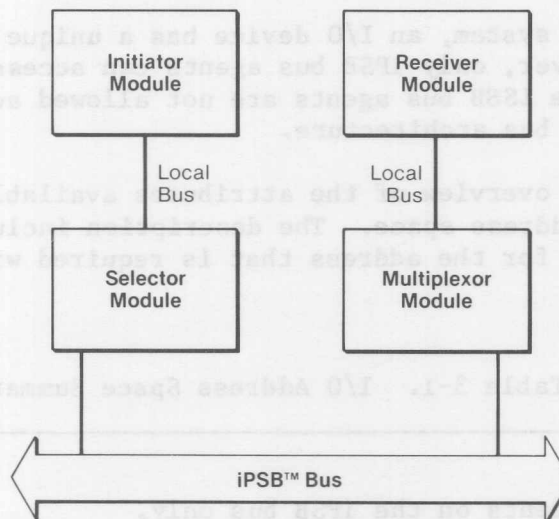


Figure 3-1. I/O Space Interface Model

#### 3.2.1 View From The Requesting Agent

The iPSB bus is the only bus in the architecture that supports I/O space as defined in this System Interface Specification. During an I/O space operation, the Initiator module provides an I/O space address to the Selector module. On sensing the I/O address, the Selector module converts the I/O address into the protocol required for the iPSB bus and sends the operation onto the bus. If the Initiator module cannot directly access I/O space, memory mapping can be used.

## SYSTEM INTERFACE SPECIFICATION

### 3.4.2 View From The Replying Agent

The Multiplexor module watches for its address on the iPSB bus. If the address occurs, the Multiplexor module causes the Receiver module to perform the operation at the I/O address specified by the Initiator module.

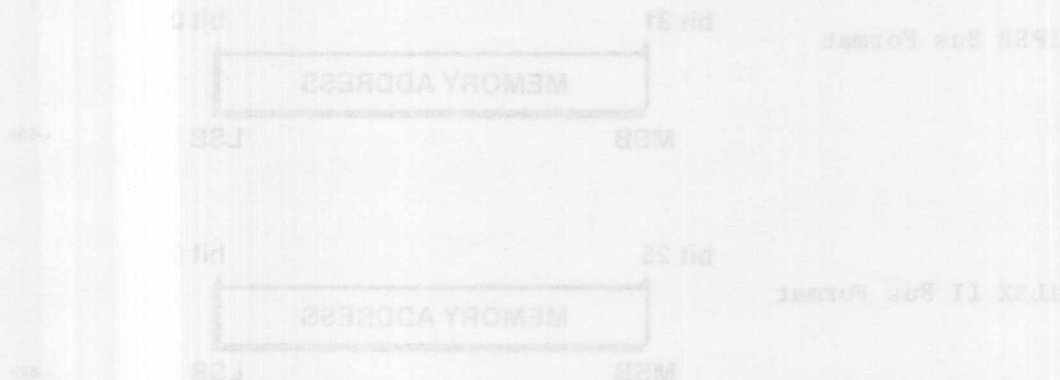
Sequential access operations are allowed to I/O space, however, the system designer must ensure that the replying agent does not increment the initial I/O address. The same I/O address is used for all data transfers during sequential access operations.

An agent accesses memory space by providing a unique address which refers to a specific memory location. Agents use the memory space to address memory modules or replying agents. Memory modules typically contain RAM devices and ROM devices, though memory mapping of other devices is possible as supported by the Initiator module.

Table 4-1 gives an overview of the addresses available for an agent operating in memory address space. The description includes listing the address format for the address required on both the iPSB bus and the iLAX II bus agents accessing the memory address space.

Table 4-1. Memory Address Space Summary

- Supports agents on the iPSB bus and on the iLAX II bus.
- Supports read and write operations.
- Supports 8-bit, 16-bit, 32-bit, and 31-bit data transfers.
- Supports sequential transfers with replying agent incrementing the address.
- Supports point-to-point operations (broadcast operations are not supported).
- Supports 31-bit addresses for iPSB bus operations and 30-bit addresses for iLAX II operations, as follows:



#### 4. MEMORY ADDRESS SPACE OPERATIONS

##### 4.1 INTRODUCTION

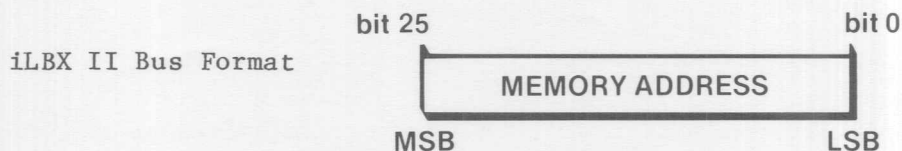
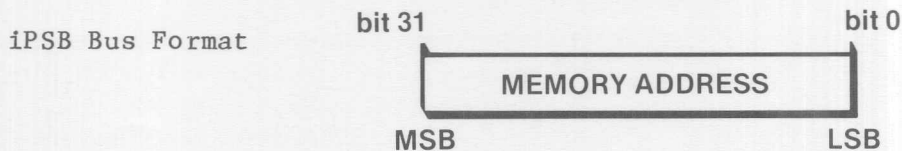
Memory space is the most familiar address space of the four defined in the Multibus II bus architecture. Agents use memory space for storing and retrieving both data and code. Operations in memory space can occur on either the iLBX II bus or the iPSB bus. Memory space operations never involve more than one replying agent.

An agent accesses memory space by providing a unique address which refers to a specific memory location. Agents use the memory space to address memory modules on replying agents. Memory modules typically consist of RAM devices and ROM devices, though memory mapping of other devices is possible if supported by the Initiator module.

Table 4-1 gives an overview of the attributes available for an agent operating in memory address space. The description includes figures showing the format for the address required on both the iPSB bus and the iLBX II bus agents accessing the memory address space.

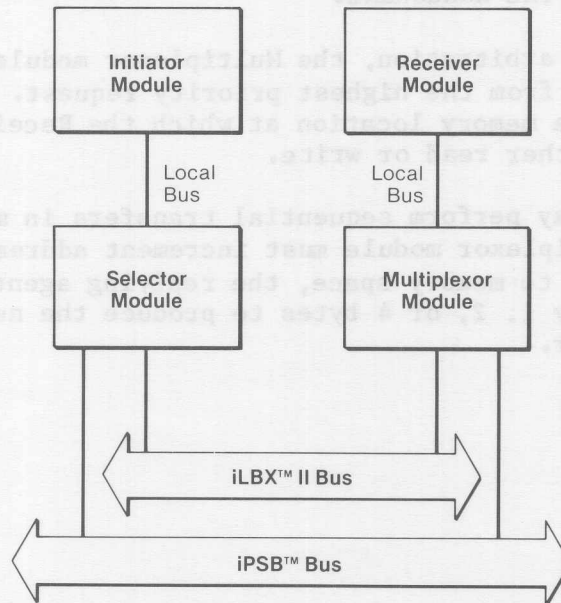
Table 4-1. Memory Address Space Summary

- o Supports agents on the iPSB bus and on the iLBX II bus.
- o Supports read and write operations.
- o Supports 8-bit, 16-bit, 24-bit, and 32-bit data transfers.
- o Supports sequential transfers with replying agent incrementing the address.
- o Supports point-to-point operations (broadcast operations are not supported).
- o Supports 32-bit addresses for iPSB bus operations and 26-bit addresses for iLBX II operations, as follows:



## 4.2 INTERFACE MODEL FOR MEMORY ADDRESS SPACE

Figure 4-1 shows the system interface model for memory space. Memory space supports varying data widths and sequential accesses, both of which must be monitored by the Multiplexor module. Operations in memory space usually consist of read and write operations in RAM devices or read operations in ROM devices.



x-590

Figure 4-1. Memory Address Space Interface Model

## 4.2.1 View From The Requesting Agent

The requesting agent consists of the Initiator module and the Selector module. The Initiator module accesses Memory space by initiating an operation that contains a memory address. That operation may involve agents on either the iLBX II bus or the iPSB bus; both support Memory space operations. The system designer must divide the memory space within the system into two non-overlapping areas; one dedicated only to the iLBX II bus and one dedicated only to the iPSB bus.

The selector module in the requesting agent is responsible for ensuring that agents do not send one operation onto both the iLBX II bus and the iPSB bus. The Selector module interprets the address associated with each operation and selects the appropriate bus structure for the operation.



## SYSTEM INTERFACE SPECIFICATION

### 4.2.2 View From The Replying Agent

If a replying agent includes an interface to both the iPSB bus and the iLBX II bus, the Multiplexor module monitors for its address on both busses.

If two different operations (one on the iPSB bus and one on the iLBX II bus) arrive at the replying agent simultaneously, the Multiplexor module arbitrates between the two requests. As a result of operation of the arbitration algorithm in the Multiplexor module, the higher priority access gains service; the lower priority access is postponed by delaying the completion of the handshake.

On completing the arbitration, the Multiplexor module gives the Receiver module an address from the highest priority request. That memory address refers to a unique memory location at which the Receiver module performs the operation, either read or write.

Replying agents may perform sequential transfers in memory space; however, the Multiplexor module must increment address. During a sequential access to memory space, the replying agent increments the initial address by 1, 2, or 4 bytes to produce the new address for the next data transfer.

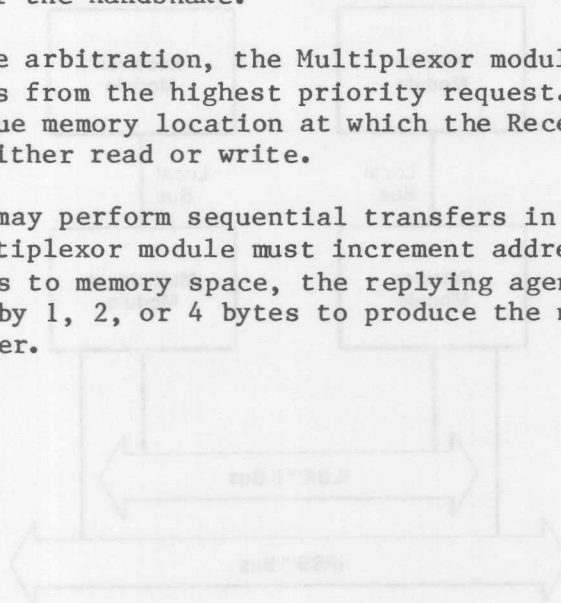


Figure 4-1. Memory Address Space Interface Model

### 4.2.1 View From The Requesting Agent

The requesting agent consists of the Initiator module and the Selector module. The Initiator module accesses memory space by initiating an operation that contains a memory address. That operation may involve agents on either the iPSB II bus or the iLBX II bus; both support memory space operations. The system designer must divide the memory space into two non-overlapping areas; one dedicated only to the iPSB bus and one dedicated only to the iLBX II bus.

The Selector module in the requesting agent is responsible for ensuring that agents do not send one operation onto both the iPSB II bus and the iLBX II bus. The Selector module interprets the address associated with each operation and selects the appropriate bus structure for the operation.

## 5. MESSAGE SPACE OPERATIONS

## 5.1 INTRODUCTION

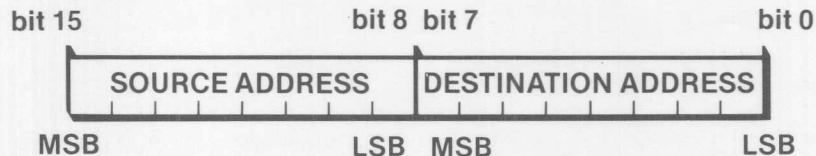
The Multibus II bus architecture uses message address space for the implementation of multiple processor architectures that require interprocessor communication. Message address space is accessible to agents on the iPSB bus and iSSB bus.

Agents utilizing the message address space are assigned a message space address. This address is used by the agent to recognize messages routed to it. In addition, one address (OFFH) is designated as a broadcast address which all agents recognize. This addressing scheme allows any agent to send a message to any other agent without regard for the receiving agent's local environment. This decoupling effect significantly simplifies the software required to perform interprocessor communication in the open system environment of the Multibus II bus architecture.

Table 5-1 gives an overview of the attributes available for an agent operating in message address space. These attributes apply to both the iPSB bus and iSSB bus agents.

Table 5-1. Message Address Space Summary

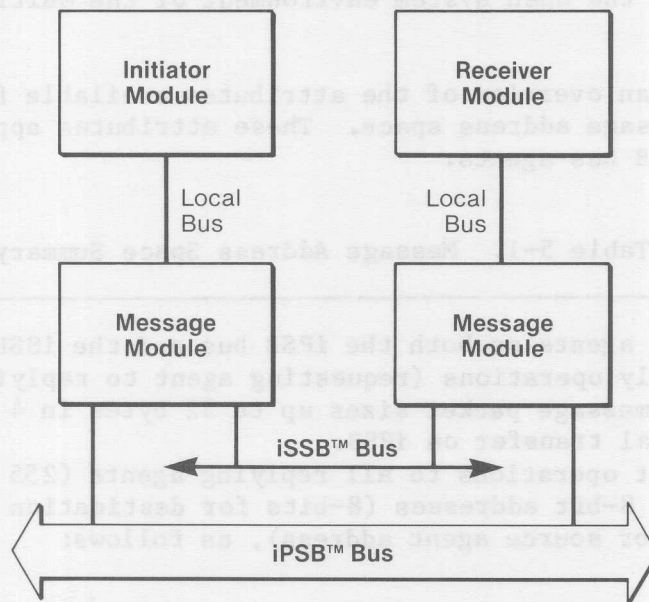
- o Supports agents on both the iPSB bus and the iSSB bus.
- o Write only operations (requesting agent to replying agent).
- o Varying message packet sizes up to 32 bytes in 4 byte increments.
- o Sequential transfer on iPSB.
- o Broadcast operations to all replying agents (255 maximum).
- o Supports 8-bit addresses (8-bits for destination agent address, 8-bits for source agent address), as follows:



x-538

## 5.2 INTERFACE MODEL

Figure 5-1 shows the message space interface model. The only change over the previous interface model is to replace the selector module and multiplexer module with more capable message modules. In addition to the added capability, the message modules retain the selector module and multiplexer module functions. The initiator module and receiver module remain unchanged. The following sections describe the function of the message modules for the local bus interface and the iSSB/iPSB bus interface.



x-602

Figure 5-1. Message Space Interface Model

## SYSTEM INTERFACE SPECIFICATION

### 5.2.1 Local Bus Interface

The message module is responsible for providing a highly capable interface to the initiator module and receiver module. This interface is intended to enhance the performance of an agent by decoupling the local bus from the iPSB bus and iSSB bus, and by off-loading time consuming data copy operations. This interface is not part of this specification. It is only described at a high level as needed to clearly specify the message address space of the iPSB bus and iSSB bus.

One possible implementation of this interface would be to use a FIFO structure (either in shared memory or in the message module) to pass short messages over the local bus. These messages could be used to either directly send a message (unsolicited message) or to control a larger data transfer (solicited message). For larger data transfers, DMA controllers in the message modules would be used to move the data block. The message modules would also contain sufficient buffering to decouple the operation of the local bus from the iPSB bus or iSSB bus.

### 5.2.2 iSSB™/iPSB™ Interface

The message module is responsible for providing a compatible interface to the iSSB bus and/or the iPSB bus message address space. Operations in message address space consist of transferring packets between agents. A packet is a group of 4 to 32 data bytes which comprise a message or fragment of a message. Detailed packet formats are provided in section 5.6 of this specification.

The use of packets to transfer messages on the iSSB bus and iPSB bus provides two benefits. First, packets limit the time any one agent may hold the bus, allowing a worst case access time to be calculated (essential for real time applications). Second, bounded length packets can be easily buffered in the message module. This allows transfers on the iPSB bus and the iSSB bus to be decoupled from the local bus on both agents. This enhances system performance by maximizing the available bandwidth of all buses in the system.

Packets on the iPSB bus and iSSB bus carry the same logical information. The only difference between the message address spaces on the two buses is the encapsulation and transfer protocol (e.g. The iPSB bus uses byte parity and sequential transfers). Further details are provided in the respective bus specifications.

Typical packet formats for the iPSB bus and the iSSB bus are shown in Figures 5-2 and 5-3 respectively. Each contains the same five fields: destination address, source address, type, type specific, and data. Each is described in more detail in the following paragraphs.

The destination address field identifies the agent where the receiver module exists. This field identifies an unique receiver module (and agent) with values 0-OFEH. The value OFFH identifies a broadcast to all agents. This field is required in all packets.



## SYSTEM INTERFACE SPECIFICATION

The source address field identifies the initiator module (and agent). This field may only contain values between 0 and OFEH. This field is required in all packets.

The type field identifies the further fields in the packet. This specification defines six types and their subsequent fields. All other types are reserved. This field is required in all packets.

The type specific field is an additional control field used for some packet types. Further definition of this field is provided later in this specification. When this field is labeled "reserved", it may not be used and its contents are not guaranteed. This field is required in all packets.

The data field definition is based on the type field. Details are provided later in this specification. In general this field is variable in length from 0 to 28 bytes in 4 byte increments.

Address/Data Lines				
	AD31★ to AD24★	AD23★ to AD16★	AD15★ to AD8★	AD7★ to AD0★
Request Phase			Source Address Byte 2	Destination Address Byte 1
Handshake 1			Type Specific Byte 4	Type Byte 3
Handshake 2	Data Byte Byte 8	Data Byte Byte 7	Data Byte Byte 6	Data Byte Byte 5
Handshake 3	Data Byte Byte 12	Data Byte Byte 11	Data Byte Byte 10	Data Byte Byte 9
Handshake 4	Data Byte Byte 16	Data Byte Byte 15	Data Byte Byte 14	Data Byte Byte 13
Handshake 5	Data Byte Byte 20	Data Byte Byte 19	Data Byte Byte 18	Data Byte Byte 17
Handshake 6	Data Byte Byte 24	Data Byte Byte 23	Data Byte Byte 22	Data Byte Byte 21
Handshake 7	Data Byte Byte 28	Data Byte Byte 27	Data Byte Byte 26	Data Byte Byte 25
Handshake 8	Data Byte Byte 32	Data Byte Byte 31	Data Byte Byte 30	Data Byte Byte 29

x-589

Figure 5-2. Typical iPSB™ Packet

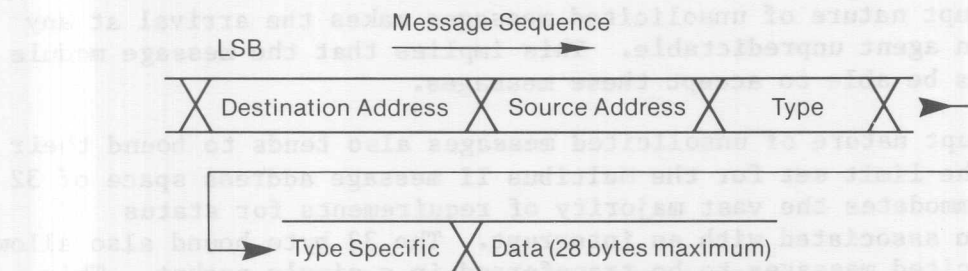


Figure 5-3. Typical iSSB™ Packet

### 5.3 MESSAGE MODEL

The message address space is used in the Multibus II bus architecture for interprocessor communication between modules on different agents. This section presents a basic model for this communication mechanism which is used in conjunction with the interface model to further define message address space operation.

All messages used for interprocessor communication can be classified as "interrupt-like" or "data-like." This specification defines these as unsolicited messages and solicited messages respectively.

#### 5.3.1 Unsolicited Messages

Unsolicited messages may be viewed as "intelligent interrupts" within the Multibus II bus architecture. These messages provide up to 255 interrupt sources (the number of valid source addresses) and allow status information to be directly attached via the data field. Attaching the status information in the data field eliminates the need for additional operations prior to servicing the interrupt.

## SYSTEM INTERFACE SPECIFICATION

**5.3.1.1 CHARACTERISTICS OF UNSOLICITED MESSAGES.** The general characteristics of an unsolicited message are:

- o Unpredictable arrival
- o Bounded length
- o Require bounded delivery time

The interrupt nature of unsolicited messages makes the arrival at any destination agent unpredictable. This implies that the message module must always be able to accept these messages.

The interrupt nature of unsolicited messages also tends to bound their length. The limit set for the Multibus II message address space of 32 bytes accommodates the vast majority of requirements for status information associated with an interrupt. The 32 byte bound also allows all unsolicited messages to be transferred in a single packet. This simplifies the problem of unpredictable arrival by simply requiring a message module to be capable of receiving and buffering one or more packets.

The interrupt nature of unsolicited messages also adds a "real time" (i.e., response required in a bounded time period) requirement to their delivery. Even with careful functional partitioning, where the most time critical events are confined to a single agent, there will still be some less demanding time critical operations between agents. In message space, the bounded packet size and deterministic access protocols for the iPSB bus and iSSB bus are designed to meet the needs of these systems.

**5.3.1.2 UNSOLICITED MESSAGE TYPES.** This specification defines the following five unsolicited message types:

- o Unsolicited output message
- o Unsolicited output broadcast message
- o Reset message
- o Buffer request message
- o Buffer grant message

The following paragraphs in this section provide a brief description of each type. Detailed packet formats are presented later in Section 5.6.

The unsolicited output message may be viewed as the generic interrupt. This message allows an initiator module to interrupt a receiver module and simultaneously provide up to 28 bytes of user defined status information. If the data field is not used, this message functions like a standard interrupt signal line, providing an interrupt signal and source vector (i.e. source address). The data field allows for enhanced performance by eliminating the need for the receiver module to poll the initiator module via memory, I/O or interconnect address space for the status information.

## SYSTEM INTERFACE SPECIFICATION

The unsolicited output broadcast message is logically identical to the unsolicited output message. The only difference is that the unsolicited output broadcast message is received by all agents in message address space. Note that a broadcast on the iPSB bus or iSSB bus disables the handshake protocol (i.e. acknowledge) and therefore, reception of a broadcast is not guaranteed without a higher level message exchange. Typical uses for broadcast are requests for a resource or repetitive signalling (e.g. time of day clock, watchdog timer signals, etc.).

The reset message is a special unsolicited message type that allows an initiator module to reset another agent in the system. This message carries no data field. The reset operation is performed via a hardware signal from the receiver's message module. This message never directly reaches the receiver module.

The buffer request and buffer grant messages are unsolicited messages used to set up a solicited message transfer. These messages have predefined data fields which are described in detail in Sections 5.5 and 5.6. The buffer request message is used to set up a DMA controller in the initiator's message module and to request a buffer from the receiver module. The buffer grant message is used to set up a DMA controller in the receiver's message module and to inform the initiator's message module that the transfer can begin.

**5.3.1.3 LOCAL MESSAGES.** Local messages are unsolicited messages used only for communication between the initiator module or receiver module and their respective message modules. These messages do not pass across the iPSB bus or iSSB bus and, therefore, are not part of this specification. They are only mentioned in this specification where needed for clarity. Specific implementations of the message module can define these messages as required.

### 5.3.2 Solicited Messages

Solicited messages are used to move blocks of data between agents in the Multibus II bus architecture. These messages provide the capability of moving data with minimum involvement by the initiator module and receiver module.

**5.3.2.1 CHARACTERISTICS OF SOLICITED MESSAGES.** The general characteristics of a solicited message are:

- o Data intensive (up to 64k bytes)
- o Arrival is negotiable

The data intensive (i.e. no "reasonable bound") nature of solicited transfers prevents them from being handled as unsolicited messages are. It is impractical to require agents to provide buffering for large quantities of data whose arrival is unpredictable.



## SYSTEM INTERFACE SPECIFICATION

Fortunately, the nature of solicited messages is such that their transfer is always negotiable. For example, a file access on a disk is not something that happens spontaneously; rather, it is a predictable event that is planned for by the receiver of the data. For the message address space, the negotiation process is done with unsolicited messages.

**5.3.2.2 DATA PACKETS.** Solicited messages cannot be transferred in a single packet like an unsolicited message. Since the message address space limits all transfers to 32 bytes or less, it is necessary to define a data packet for transferring solicited messages.

The data packet follows the standard packet format shown in Figures 5-2 and 5-3. The data field is used to carry fragments of the solicited message. The initiator's message module is responsible for creating these packets and the receiver's message module is responsible for reassembling them. A solicited message may require one or more data packets depending on its length. A detailed packet format is presented in Section 5.6. Further description of the control fields is presented in Section 5.5.

**5.3.2.3 LIAISON ID.** The need for multiple packets in a solicited operation leads to a problem in identifying the packets at the message module. In order to permit multiple solicited operations, these packets contain an identification number referred to as a liaison ID. The liaison ID is a number assigned by a message module for binding (associating) a response packet with one it is sending. More details are provided in the solicited message transfer discussion in section 5.5.

**5.3.2.4 DUTY CYCLE.** The high speed of the iPSB bus compared to local buses creates a potential problem for solicited transfers. If a 32 bit initiator module is sending a solicited message to a 16 bit receiver module, the receiver module may not be able to service packets as fast as they arrive, assuming they are sent back to back. The result of this condition would be many NACKs on the iPSB bus which reduces system performance. The duty cycle parameter for a solicited transfer is provided to minimize this problem. This value, assigned by the receiver module, specifies the required delay, in microseconds, between transmitted packets needed to guarantee that the message module is not overrun.

# SYSTEM INTERFACE SPECIFICATION

## 5.4 UNSOLICITED MESSAGE TRANSFER

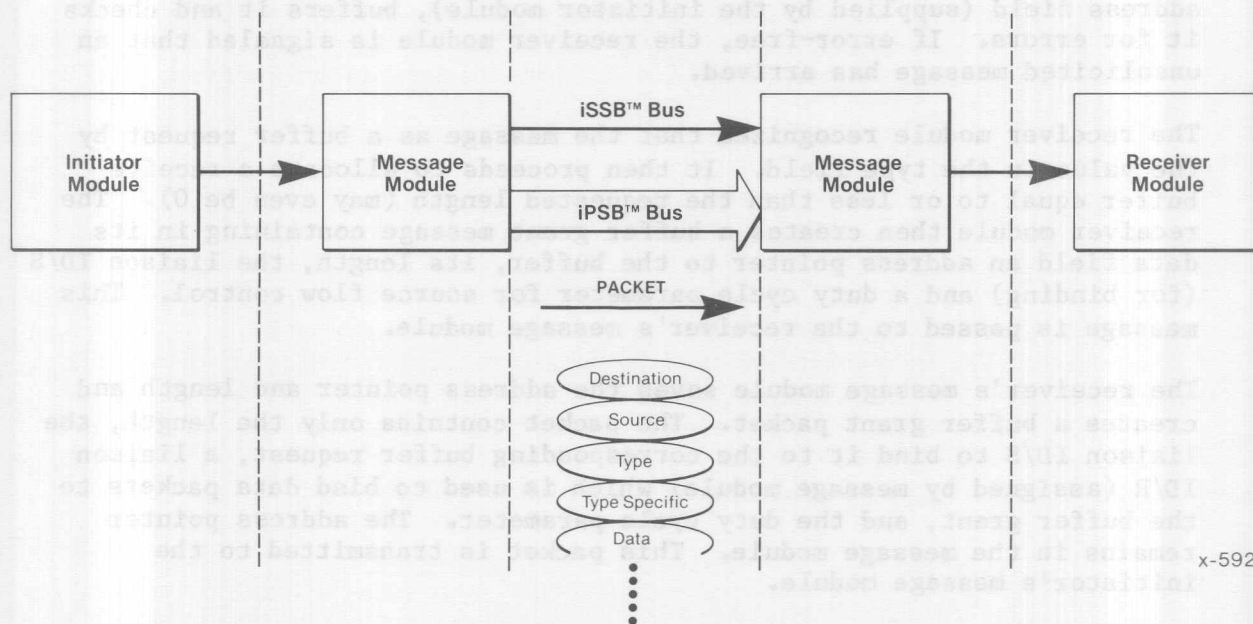
A typical unsolicited message transfer is shown in figure 5-4. The following paragraphs describe the transfer process in more detail.

The initiator module is responsible for generating the unsolicited message. The transfer begins when the initiator module passes the message to its message module.

The initiator's message module is responsible for creating a packet from the message. Depending on the message type, this may involve no specific action (e.g. unsolicited output message). The initiator's message module then transmits the packet to the receiver's message module.

The receiver's message module is responsible for monitoring message address space for a valid address (i.e. its receiver module's address or the broadcast address). If a valid address is recognized, the packet is buffered and checked for errors. If the packet is error-free, the receiver's message module signals the receiver module that a message has arrived.

Upon being signaled that a message has arrived, the receiver module interprets and processes it as it would any other interrupt.



x-592

VI

Figure 5-4. Typical Unsolicited Transfer

## SYSTEM INTERFACE SPECIFICATION

### 5.5 SOLICITED MESSAGE TRANSFER

A typical solicited message transfer is shown in figure 5-5. This transfer can be divided into three phases: the negotiation, the transfer, and completion.

#### 5.5.1 Negotiation Phase

Solicited messages must negotiate for buffer space before a transfer can take place. This negotiation is performed by a pair of unsolicited messages: the buffer request and the buffer grant.

The initiator module begins a solicited operation by passing a buffer request message to its message module. The data field of the buffer request message contains an address pointer to the solicited message and its length. The address pointer is a physical address into the initiator module's memory.

The initiator's message module saves the address pointer and creates a buffer request packet. The data field of the packet contains only the length and a liaison ID/S (assigned by message module) which is used to bind the buffer request message to the corresponding buffer grant message. The address pointer remains in the message module. This packet is transmitted to the receiver's message module.

The receiver's message module recognizes the packet by its destination address field (supplied by the initiator module), buffers it and checks it for errors. If error-free, the receiver module is signaled that an unsolicited message has arrived.

The receiver module recognizes that the message as a buffer request by the value in the type field. It then proceeds to allocate a receive buffer equal to or less than the requested length (may even be 0). The receiver module then creates a buffer grant message containing in its data field an address pointer to the buffer, its length, the liaison ID/S (for binding) and a duty cycle parameter for source flow control. This message is passed to the receiver's message module.

The receiver's message module saves the address pointer and length and creates a buffer grant packet. The packet contains only the length, the liaison ID/S to bind it to the corresponding buffer request, a liaison ID/R (assigned by message module) which is used to bind data packets to the buffer grant, and the duty cycle parameter. The address pointer remains in the message module. This packet is transmitted to the initiator's message module.

The initiator's message module recognizes the packet by its destination address (supplied by the receiver module), buffers it and checks it for errors. If error-free, the message module recognizes the type and liaison ID/S fields and completes the negotiation phase by saving the length, liaison ID/R and duty cycle parameters. This packet is not sent to the initiator module.



## SYSTEM INTERFACE SPECIFICATION

### 5.5.2 Transfer Phase

Upon completion of the negotiation phase, the transfer phase is automatically initiated. In this phase, the solicited message is transferred from the initiator module's buffer to the receiver module's buffer by the respective message modules using a series of data packets. The initiator module and receiver module do not participate in this phase.

The initiator's message module builds data packets by directly accessing the initiator module's buffer. These packets consist of up to 28 bytes of data (solicited message fragment) and the liaison ID/R to bind the packet to a buffer at the receiver's message device. These packets are sent to the receiver's message module at a rate specified by the duty cycle parameter.

The receiver's message device recognizes the packet by its destination address (supplied by the initiator's message module) buffers it and checks it for errors. If error-free, the message device puts the data directly into the receiver module's buffer.

### 5.5.3 Completion Phase

The completion phase occurs immediately after the last data packet has been transferred or an irrecoverable error occurs (recoverable errors are dealt with in the iPSB bus and iSSB bus specifications). During this phase there are no packets transferred on the iPSB bus or iSSB bus. All activity is either within the message modules or on the message module's local buses.

During the completion phase, the message modules clear any internal state related to the transfer and send a local message to the initiator module and receiver module. This local message is referred to as a general completion message and is used to indicate the status of the transfer on termination.



# SYSTEM INTERFACE SPECIFICATION

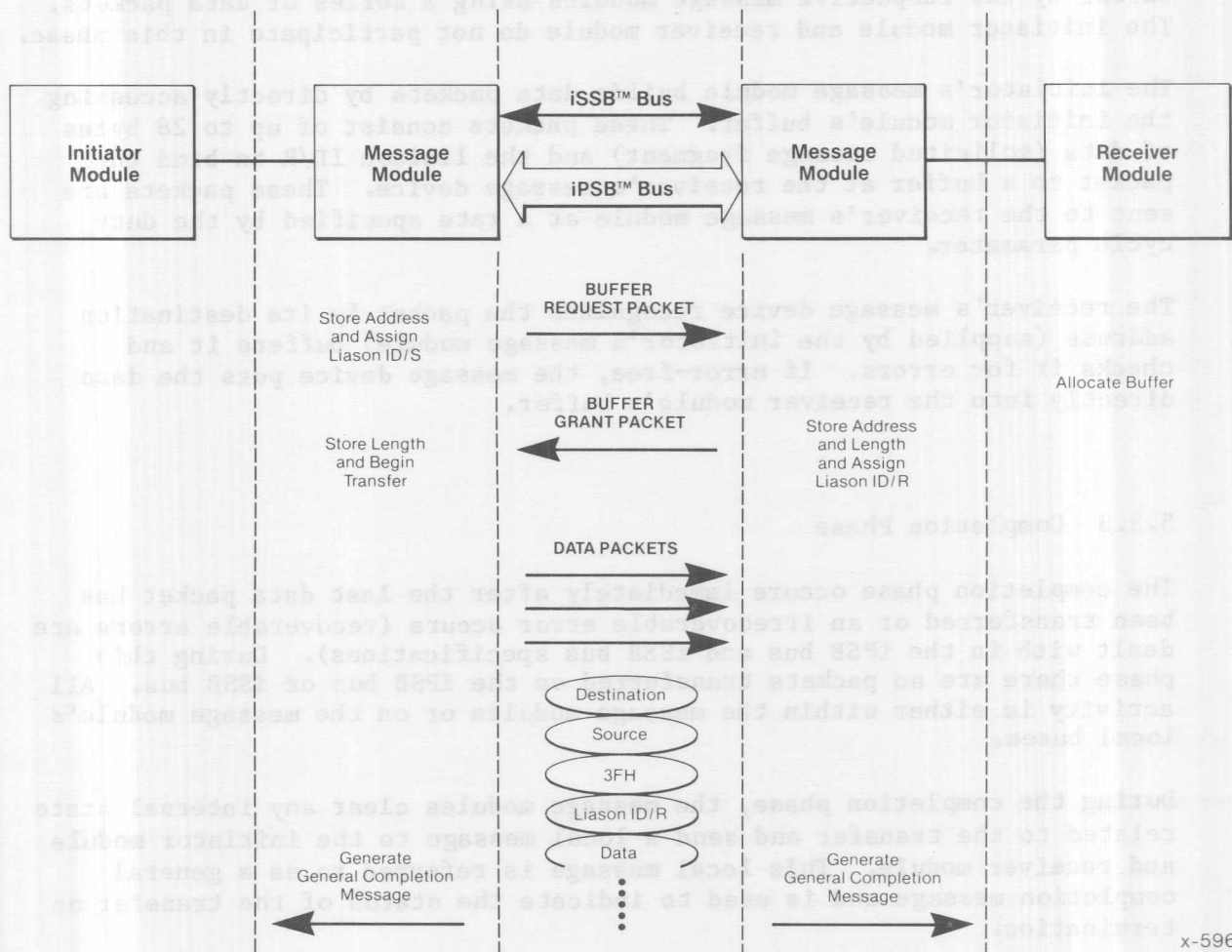


Figure 5-5. Typical Solicited Message Transfer

# SYSTEM INTERFACE SPECIFICATION

## 5.6 PACKET FORMATS

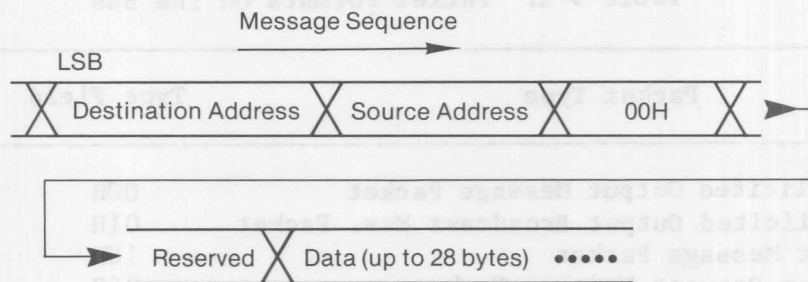
This section specifies the packet formats for the message space. Table 5-2 summarizes the packets defined in this specification, listing their type and referencing a figure containing the format for the iPSB bus and iSSB bus. Note that in the figures the iPSB bus format is shown only for the 16-bit transfer. Details of the 32-bit transfers are provided in the iPSB Bus Specification.

Table 5-2. Packet Formats On The Bus

Packet Type	Type Field	Figure
Unsolicited Output Message Packet	00H	5-6
Unsolicited Output Broadcast Mes. Packet	01H	5-7
Reset Message Packet	10H	5-8
Buffer Request Message Packet	24H	5-9
Buffer Grant Message Packet	35H	5-10
Data Packet	3FH	5-11

# SYSTEM INTERFACE SPECIFICATION

## On the iSSB™ Bus:



## On the iPSB™ Bus:

Address-Data Lines	AD15★ to AD8★	AD7★ to AD0★
Request Phase	Source Address	Destination Address
Handshake 1	Reserved	00H
Handshake 2	Subsequent Transfers are 16-bits or 32-bits wide	
...		

x-584

Figure 5-6. Unsolicited Output Message Packet

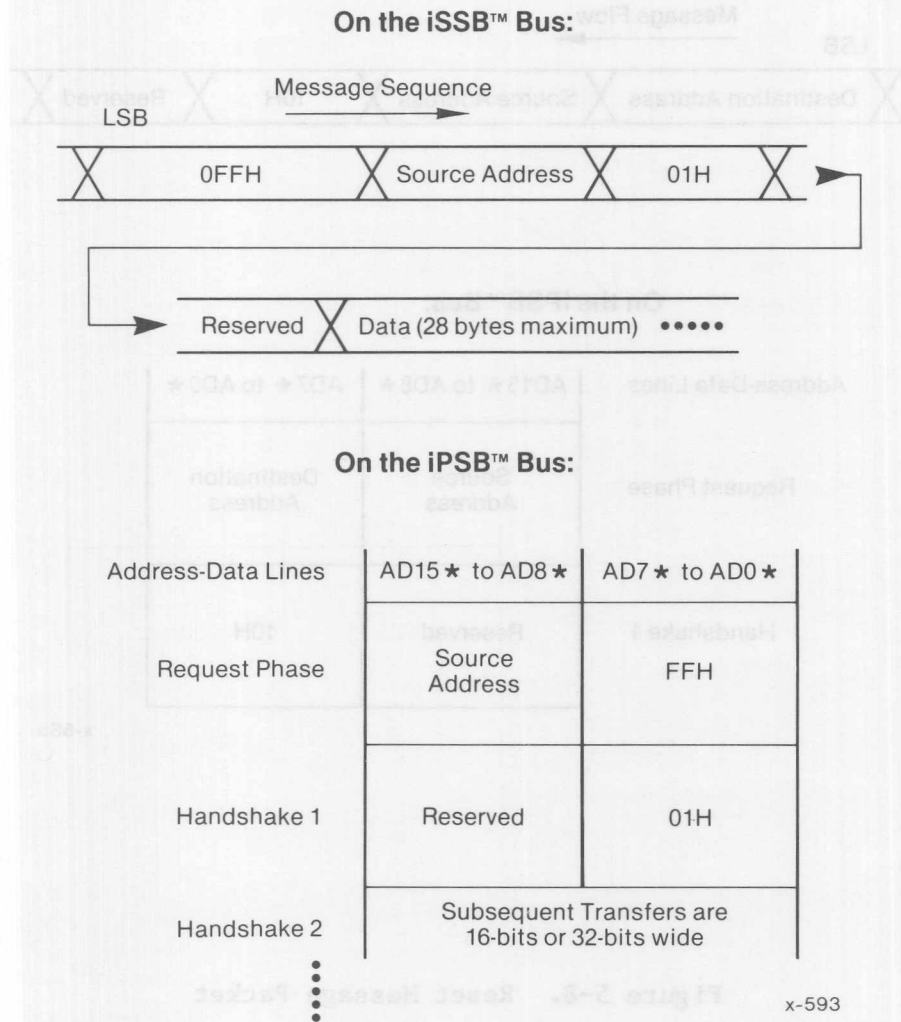
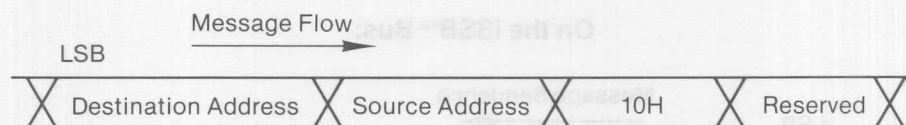


Figure 5-7. Unsolicited Output Broadcast Message Packet



On the iSSB™ Bus:



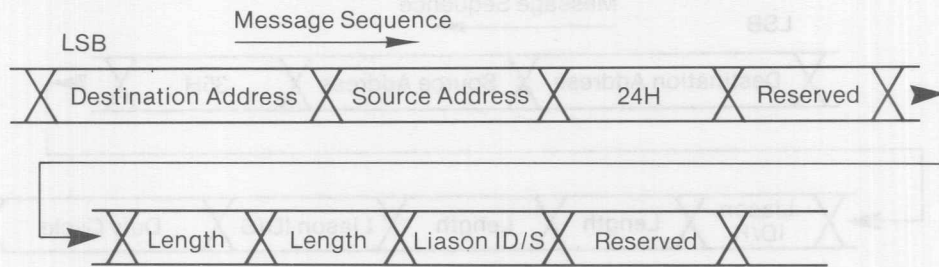
On the iPSB™ Bus:

Address-Data Lines	AD15 ★ to AD8 ★	AD7 ★ to AD0 ★
Request Phase	Source Address	Destination Address
Handshake 1	Reserved	10H

x-585

Figure 5-8. Reset Message Packet

On the iSSB™ Bus:

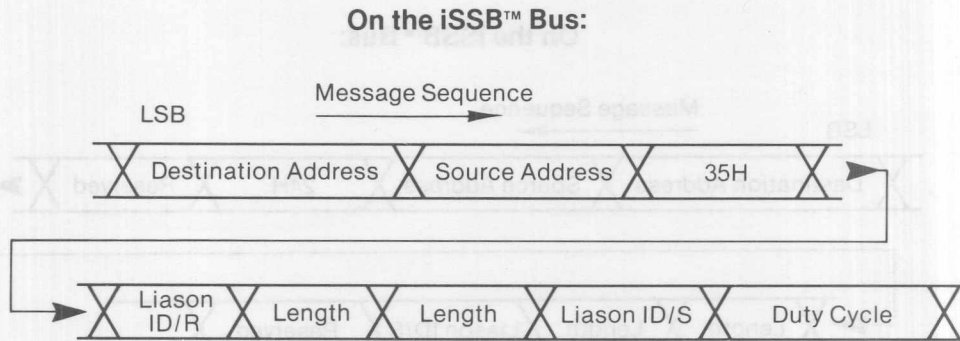


On the iPSB™ Bus:

Address-Data Lines	AD15★ to AD8★	AD7★ to AD0★
Request Phase	Source Address	Destination Address
Handshake 1	Reserved	24H
Handshake 2	Length	
Handshake 3	Reserved	Liason ID/S

x-594

Figure 5-9. Buffer Request Message Packet



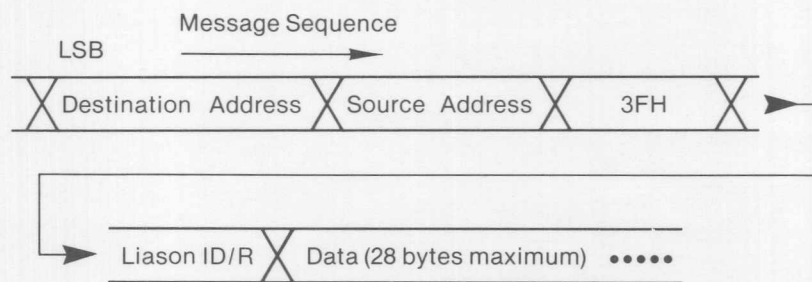
**On the iPSB™ Bus:**

Address-Data Lines	AD15* to AD8*	AD7* to AD0*
Request Phase	Source Address	Destination Address
Handshake 1	Liason ID/R	35H
Handshake 2	Length	
Handshake 3	Duty Cycle	Liason ID/S

x-587

Figure 5-10. Buffer Grant Message Packet

On the iSSB™ Bus:



On the iPSB™ Bus:

Address-Data Lines	AD15★ to AD8★	AD7★ to AD0★
Request Phase	Source Address	Destination Address
Handshake 1	Liason ID/R	3FH
Handshake 2	Subsequent Transfers are 16-bits or 32 bits wide	
⋮		

x-622

Figure 5-11. Data Packet

\*\*\*



